UNIVERSITY OF CALIFORNIA
RIVERSIDE

Solution Path Clustering with Robust Loss and Concave Penalty

A Dissertation submitted in partial satisfaction
of the requirements for the degree of

Doctor of Philosophy

in

Applied Statistics

by

Edward Schuberg

March 2019

Dissertation Committee:

Dr. Weixin Yao, Co-Chairperson
Dr. Shujie Ma, Co-Chairperson
Dr. Wenxiu Ma

The Dissertation of Edward Schuberg is approved:

Co-Chairperson

Co-Chairperson

University of California, Riverside

Acknowledgments

I am grateful to my advisors, Dr. Weixin Yao and Dr. Shujie Ma. Their patient guidance has been invaluable, with many important lessons both inside and outside the world of statistics. I would also like to thank Dr. Wenxiu Ma for her time, support, and good will throughout the committee process. I am grateful to Dr. Daniel Jeske whose mentorship and outreach completely changed the trajectory of my professional life. Finally, I am grateful to my undergraduate advisor, Dr. David Andrews, without whose encouragement I would not be here. Encouragement is a hard thing to overestimate.

To my wife.

ABSTRACT OF THE DISSERTATION

Solution Path Clustering with Robust Loss and Concave Penalty

by

Edward Schuberg

Doctor of Philosophy, Graduate Program in Applied Statistics
University of California, Riverside, March 2019
Dr. Weixin Yao, Co-Chairperson
Dr. Shujie Ma, Co-Chairperson

The main purpose of this dissertation is to demonstrate that using a robust loss function (instead of the usual least squares loss) improves the clustering quality in the solution path clustering scheme. Cluster analysis simultaneously attempts to determine the number of clusters and estimate cluster location and membership. Convex clustering, distinguishing itself from other popular clustering methods, casts the clustering objective as a convex optimization problem and thus admits a global solution. It is a useful exploratory technique which outputs a solution path, evoking the name, "solution path clustering." The solution path is a tree-like structure with cluster results ranging from n clusters down to a single cluster. Now, the benefits of convex clustering come at a cost since the use of a convex penalty can seriously bias the results and ruin the search for good cluster results. To lessen the bias, Ma and Huang (2017) proposed concave penalties to form the cluster centers. While the clustering objective is no longer convex, the quality of the solutions is improved.

We extend the solution path clustering scheme by implementing robust loss func-

tions instead of the usual least squares loss. Following Ma and Huang (2017), we also use a concave penalty to form clusters. The robust loss and concave penalty work together to mitigate the influence of outliers and minimize bias in the estimation of cluster locations, especially when the true distance between clusters is large. We introduce the IRLS-ADMM algorithm to minimize our proposed objective function and prove its convergence to a local minimum. Any loss function that admits an IRLS formulation or a majorizing surrogate can be used. We also study asymptotic and oracle properties of the estimator. Finally, we demonstrate the performance of our proposed method through simulation experiments and on real data sets, as well as provide some preliminary results on choosing the number of clusters via the modified BIC (Wang, Li, and Leng, 2009).

Contents

List of Figures

List of Tables

Chapter 1

Introduction to Clustering

Clustering is a fundamental and challenging problem in many applications. The goal is to partition the data into homogeneous subgroups in a relevant and meaningful way without any prior knowledge of the structure of the data. This makes cluster analysis a type of unsupervised learning. The data is believed to be coming from a population that is composed of several distinct sub-populations which we wish to detect. Clustering methods are comprised of fundamental techniques used in statistics, machine learning, pattern recognition, as well as in many applied fields. Understood loosely, clustering is essentially "the practice of classifying objects according to perceived similarities [and] is the basis for much of science" (Jain and Dubes, 1988).

There are many books on clustering, such as Hartigan (1975), Jain and Dubes (1988), Kaufman and Rousseeuw (1990), Gordon (1999), and Everitt et. al. (2011). Xu and Wunsch (2005) also provide an excellent survey of clustering algorithms. Clustering also has many applications in a variety of fields. Banerjee and Ghosh (2001) use cluster analysis

on click-stream patterns on websites to generate customized user content. Tzanetakis and Cook (2002) use clustering for the classification of musical genres. Dolan and Van der Maas (1998) discuss mixture models in the structural equation modeling context. Ma and Huang (2016) discuss subgroup analysis for personalized medicine and treatment decisions. Schork, Allison, and Thiel (1996) discuss mixture models to assess the impact of possible underlying genotypes, where order selection is key in the identification of the existence of major genes and whether or not they are dominant. Chen and Maitra (2011) discuss cluster analysis with an application to mutual funds.

Interestingly, the human eye can quite naturally and rapidly identify groupings and clusters in a two dimensional scatter plot. However, the need to systematically define how to form clusters and how many clusters to form becomes apparent in order to generalize to higher dimensional data. This also solves the problem when different people identify (justifiably) different groupings in the data. In Everitt (1974), a few definitions of a cluster are offered:

1. "A cluster is a set of entities which are *alike*, and entities from different clusters are not alike."

2. "A cluster is an aggregation of points in the test space such that the *distance* between any two points in the cluster is less than the distance between any point in the cluster and any point not in it."

3. "Clusters may be described as connected regions of a multi-dimensional space containing a relatively *high density* of points, separated from other such regions by a region containing a relatively low density of points."

These three definitions offer a guide in specifying mathematically the clustering objective. For example, one might classify a set of entities to be "alike" if the "distance" (such as Euclidean distance) between two points is small enough (below some threshold).

Ultimately, cluster analysis must address two fundamental questions:

Question 1. How do we form the clusters?

Question 2. How many clusters is optimal?

Question 2 is challenging in its own right, but it also very much hinges upon a good answer to Question 1. In other words, a sensible clustering method (e.g. an appropriately specified mixture of normals) will help answer Question 2. On the other hand, if a clustering method performs poorly, any number clusters, even if optimal, will not yield a sensible partition of the data. The issue become even more mystifying since it is unclear how to define what the "optimal" number of clusters should be. It also depends on what the goal is for performing the cluster analysis. For example, the goal may be to confirm that a certain grouping exists among a set of observations, or the cluster partition may become the grounds for the design of an experiment.

Solution Path Clustering, as opposed to K-means clustering or model-based clustering, does not require the number of components (clusters) to be specified beforehand. While this partially obviates the need to select the number of clusters, we offer a selection method using modified BIC in an attempt to address this difficult issue. Thus, this project seeks to contribute to the literature which tries to resolve the two intimately linked clustering principles enumerated in Questions 1 and 2.

The rest of this chapter introduces a few of the most common clustering methods.

It reviews K-means clustering and hierarchical clustering which are, in some sense, precursors to convex clustering and Solution Path Clustering. It also reviews clustering based on finite mixture models. While mixture models present some challenges in implementation, they offer a clean theoretical framework to classify observations. They also bear some resemblance to K-means clustering. The chapter concludes with a survey of methods for selecting the number of clusters.

1.1 K-means Clustering

K-means clustering is perhaps the most well-known clustering method. It remains one of the most popular methods due to its simplicity and fast implementation. To employ the K-means methods, the user specifies beforehand the number of clusters. The objective is then to minimize the within-cluster sum of squares for that fixed number of clusters.

The following presentation of the K-means method follows closely to that of Lindsten, Ohlsson, and Ljung (2011). Let $y_1, y_2, \ldots, y_n$ be observations in $\mathbb{R}^d$. Let K be the fixed number of clusters that is specified beforehand. Let $G_j \subset \{1, 2, \ldots, n\}$ be index sets such that

$$G_j \cap G_{j'} = \emptyset \text{ for each } j \neq j' \qquad \text{and} \qquad \bigcup_{j=1}^{K} G_j = \{1, 2, \ldots, n\}$$

where $\emptyset$ denotes the empty set. Thus, $i \in G_j$ means that y_i belongs to the j-th cluster. Let $G = \{G_j\}_{j=1}^{K}$, and let $\|\cdot\|$ denote the Euclidean norm. Let card G_j denote the cardinality of G_j, that is, the number of observations in cluster j. The K-means clustering problem is

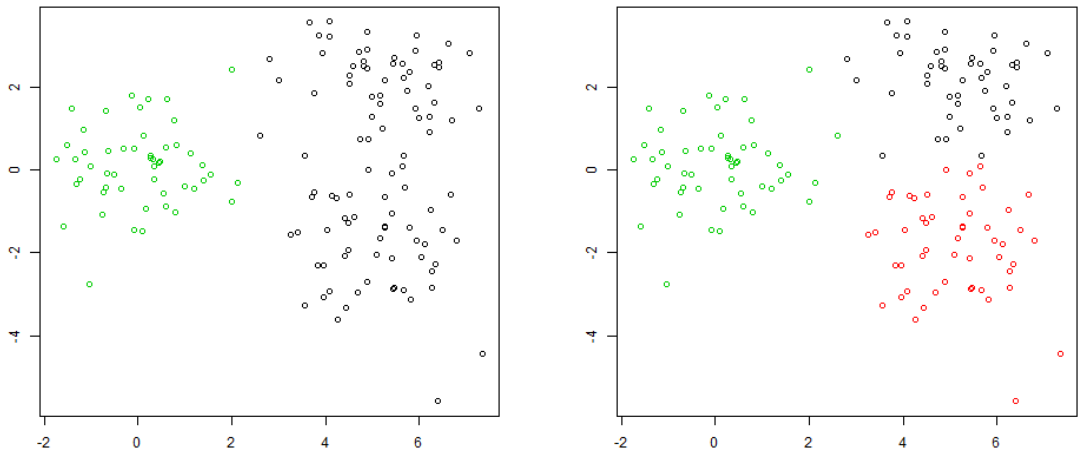
given by

$$\begin{aligned} &\min_{G} \sum_{j=1}^{K} \sum_{i \in G_j} \left\| y_i - \mu_j \right\|^2 \\ \text{such that} \quad &\mu_j = \frac{1}{\text{card } G_j} \sum_{i \in G_j} y_i \\ \text{and} \quad &\bigcup_{j=1}^{K} G_j = \{1, 2, \ldots, n\} \end{aligned} \tag{1.1}$$

Implementing an algorithm to solve this optimization problem is straightforward in most software packages. For example, the function "kmeans()" in R performs K-means with four algorithms to choose from. Multiple algorithms have been proposed to solve the problem (1.1). Though the task is quite easy to state and conceptualize, it is quite difficult to find the most optimal solution. For example, convergence to a local minimum may produce odd results. This problem is worsened by the fact that the algorithms are known to be sensitive to the initial value (Peña, Lozano, and Larraña 1999; see also Figure 1 in Lindsten, Ohlsson, and Ljung, 2011). This issue is often addressed by starting the algorithm from multiple initial values and then selecting the best solution. Another drawback to the algorithm is that it can be slow to converge (Vattani 2009). However, for the majority of datasets, the K-means algorithms are quite fast to converge and do not require many iterations.

The simplicity of the K-means method proves to be perhaps its biggest weakness. Firstly, the number of clusters, K, must be chosen and fixed beforehand. If K is chosen badly (choosing K large when it should be small, or choosing K small when it should be large), the results may be poor. Thus, practitioners often run K-means along a sequence of K values to explore different cluster solutions.

Secondly, the simplicity of the model is quite inflexible and has trouble accommodating more exotic datasets. The model implicitly assumes that the clusters are spherical and with the same variance. Thus, datasets exhibiting correlation may be difficult to cluster using the K-means method. The assumption that the clusters have the same variance implies that they are of similar size to one another. In other words, assigning observations to its closest cluster is regarded as the best assignment since it does not take into account the cluster variance. This assumption equal variances across both clusters and dimensions can be unrealistic and lead to unsatisfactory results (e.g. one could try running k-means with $k = 3$ on a Mickey mouse dataset where the ears are much smaller than the head. Part



(a) K-means with $K = 2$.

(b) K-means with $K = 3$.

Figure 1.1: There are 150 total observations with 50 generated from each of $N\big(\mu = (0,0), \Sigma = I\big)$, $N\big(\mu = (5,-2), \Sigma = I\big)$, and $N\big(\mu = (5,2), \Sigma = I\big)$. The R function "kmeans()" was used with Lloyd's algorithm.

of the head will end up getting assigned to the ears).

1.1.1 Convexifying k-means

The lack of being guaranteed the global solution to the k-means problem (1.1) motivates a convexification. If the k-means task (1.1) can be somehow relaxed to a convex problem, then the global solution can be realized. Consider first the optimization problem

$$\begin{aligned} \min_{\mu} \ & \sum_{i=1}^{n} \|y_i - \mu_i\|^2 \\ & \text{such that } \{\mu_1, \mu_2, \ldots, \mu_n\} \text{ contains } k \text{ unique vectors.} \end{aligned} \tag{1.2}$$

Hence, we shifted the problem from using k centroids to allowing a μ_i parameter for each observation y_i, so long as there are only k unique μ_i values. That an optimal solution of (1.2) corresponds to an optimal solution of (1.1) is proved in Proposition 2 of Lindsten, Ohlsson, and Ljung (2011). We now need to express the constraint in (1.2) using formal mathematical notation. Thus, we need to count the number of unique vectors in a given set $\{\mu_1, \mu_2, \ldots, \mu_n\}$.

Define δ_j to be the number of $\mu_i's$ where $\mu_j = \mu_i$, for $i < j$ and $j = 2, 3, \ldots, n$. Written mathematically, we have $\delta_j = j - 1 - \|\gamma^j\|_0$ where the vectors $\gamma^j = \left(\gamma_1^j, \gamma_2^j, \ldots, \gamma_{j=1}^j\right)$ have elements $\gamma_i^j = p(\mu_i, \mu_j)$. Here, $p(\mu_i, \mu_j)$ is a non-negative, symmetric (penalty) function that equals zero if and only if $\mu_i = \mu_j$. Consider the vector $\delta = (\delta_2, \delta_3, \ldots, \delta_n)$. Notice that $\|\delta\|_0$ gives the number of times out of $n-1$ trials that a $\mu_j, j = 2, 3, \ldots n$, was *not* unique compared to the all the $\mu_i, i = 1, 2, \ldots, j-1$. In other words, the vector μ_j is unique (compared to the μ_i that came before it) only if $\delta_j = 0$. We can thus count the number of clusters by:

1. We start with one cluster. We always have at least one cluster, which is why we only calculate δ_j starting at $j = 2, 3, \ldots, n$.

2. For $j = 2, 3, \ldots, n$, if $\delta_j = 0$, increase our count of the number of clusters by 1. If $\delta_j > 0$, leave the number of clusters unchanged.

It follows that the number of clusters (the number of unique vectors in $\{\mu_1, \mu_2, \ldots, \mu_n\}$) is $n - \|\delta\|_0$. We can hence formulate (1.2) as

$$\begin{aligned} &\min_{\mu} \sum_{i=1}^{n} \|y_i - \mu_i\|^2 \\ &\text{such that } k = n - \|\delta\|_0. \end{aligned} \tag{1.3}$$

Lindsten, Ohlsson, and Ljung (2011) state that, "In some sense, the convex function closest to the L_0 norm is the L_1 norm." They relax the k-means criterion by replacing the L_0 norm with the L_1 norm twice. Firstly, we change the constraint in (1.3) to

$$\begin{aligned} k &= n - \|\delta\|_1 \\ &= n - \sum_{j=2}^{n} |\delta_j| \\ &= n - \sum_{j=2}^{n} \left| j - 1 - \|\gamma^j\|_0 \right| \\ &= n - \sum_{j=2}^{n} j - 1 - \|\gamma^j\|_0 && \text{(since it's always non-negative)} \\ &= n - \frac{n(n-1)}{2} - \sum_{j=2}^{n} \|\gamma^j\|_0 \\ &= \frac{3n - n^2}{2} - \sum_{j=2}^{n} \|\gamma^j\|_0. \end{aligned}$$

Problem (1.3) has now been transformed to

$$\begin{aligned} &\min_{\mu} \ \sum_{i=1}^{n} \|y_i - \mu_i\|^2 \\ &\text{such that } \sum_{j=2}^{n} \|\gamma^j\|_0 = \frac{3n - n^2}{2} - k. \end{aligned} \tag{1.4}$$

We again replace the L_0 norm used here to the L_1 norm. This transforms the constraint to

$$\begin{aligned} \frac{3n - n^2}{2} - k &= \sum_{j=2}^{n} \left|\gamma^j\right| \\ &= \sum_{i<j} \left|\gamma_i^j\right| \\ &= \sum_{i<j} \left|p(\mu_i, \mu_j)\right| \\ &= \sum_{i<j} p(\mu_i, \mu_j). \qquad \text{(since the terms are non-negative)} \end{aligned}$$

If the function $p(\cdot, \cdot)$ is chosen appropriately, then we will have successfully arrived to a convex problem:

$$\begin{aligned} &\min_{\mu} \ \sum_{i=1}^{n} \|y_i - \mu_i\|^2 \\ &\text{such that } \sum_{i<j} p(\mu_i, \mu_j) = \frac{3n - n^2}{2} - k. \end{aligned} \tag{1.5}$$

For example, we may use $p(\mu_i, \mu_j) = \|\mu_i - \mu_j\|_q$ for $q \geq 1$. The final step in this development is to use the equivalent Lagrangian formulation of (1.5) which is a much more convenient optimization problem:

$$\min_{\mu} \ \sum_{i=1}^{n} \|y_i - \mu_i\|^2 + \lambda \sum_{i<j} p(\mu_i, \mu_j). \tag{1.6}$$

The book from Boyd and Vandenberghe (2004) contains a wealth information on convex analysis, including Lagrangian dual formulations. It can be shown, for example, that there

exists a $\lambda > 0$ such that (1.6) is equivalent to (1.5). However, the exact λ values are not so important since λ can be interpreted as a regularization parameter which controls the trade-off between the goodness-of-fit and the number of clusters.

1.2 Hierarchical Clustering

Hierarchical clustering, as the name suggests, builds a hierarchy of cluster solutions ranging from n clusters, where each data point is its own cluster, to a single cluster consisting of all the data points. Hierarchical clustering can be agglomerative or divisive. Agglomerative clustering begins with clusters as as singleton points which are successively merged in stages until there is only a single cluster of all points. Divisive clustering begins from the opposite direction, starting with all points in a single cluster. Clusters are then split in consecutive stages until each cluster consists of only one data point. We focus here on agglomerative clustering, following the presentation found in Hocking et. al. (2011).

Let $X \in \mathbb{R}^{n \times d}$ be the observed data matrix where the rows in X are the observations of dimension d. The agglomerative scheme for hierarchical clustering suggests the following optimization problem:

$$\begin{aligned} \min_{M \in \mathbb{R}^{n \times d}} \quad & \frac{1}{2} \big\| X - M \big\|_F^2 \\ \text{subject to} \quad & \sum_{i<j} 1_{M_i \neq M_j} \leq t. \end{aligned} \tag{1.7}$$

Here, $\big\| \cdot \big\|_F$ represents the Frobenius norm, M_i is the i-th row of M, and $1_{M_i \neq M_j} = 1$ if $M_i \neq M_j$ and equals zero otherwise. Note that $\sum_{i<j} = \sum_{i=1}^{n} \sum_{j=i+1}^{n}$ sums over the $\binom{n}{2} = n(n-1)/2$ pairs of data points. When $t \geq \binom{n}{2}$, the problem (1.7) becomes unconstrained

and $M_i = X_i$ for all i so that each data point is its own singleton cluster. When $t = \binom{n}{2} - 1$, then one pair, (M_i, M_j), will be forced to merge. This is the first step in the agglomerative scheme of hierarhical clustering. As t is sequentially decreased, more and more pairs of the clusters will be merged until finally there is only a single cluster. Note that, in general, the

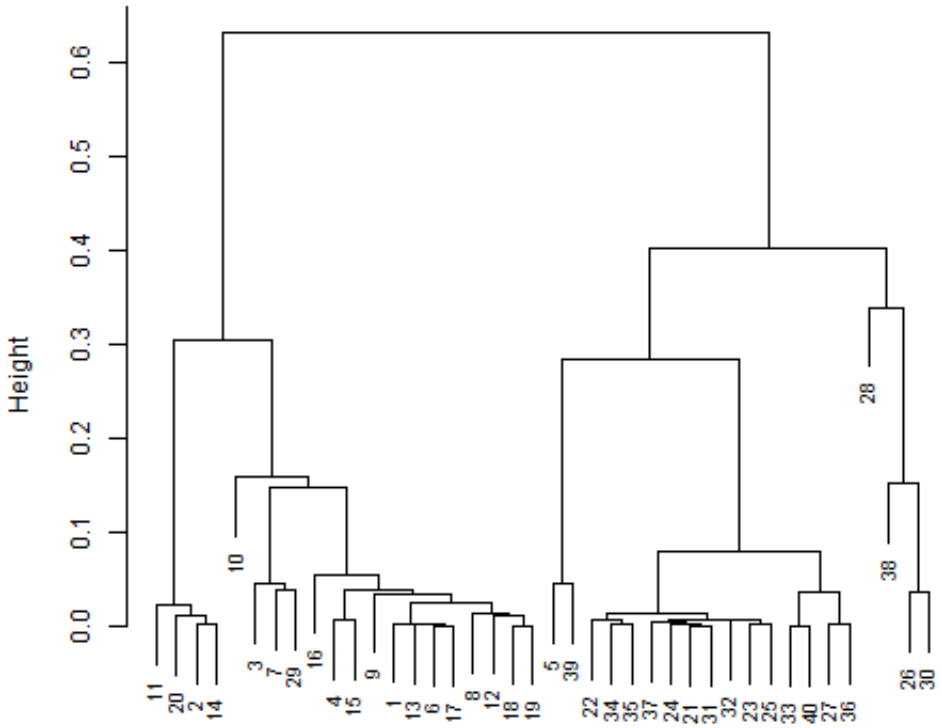


Figure 1.2: Dendrogram of an agglomerative hierarchical clustering scheme using squared Euclidean distance and single linkage. There are 40 observations with 20 generated from each of $N(\mu = 1, \sigma = 1)$ and $N(\mu = 4, \sigma = 1)$. The R function "hclust()" was used.

task of finding the best pair of clusters to merge at each step is a difficult combinatorial problem. See Murtagh and Contreras (2011) for a recent review of agglomerative clustering algorithms and how they are implemented efficiently.

Note that the problem (1.7) is only one such formulation of hierarchical clustering. There are, in fact, many options that one may choose which govern how the clustering is performed. Generally, hierarchical clustering requires the use to specify a measure of dissimilarity between sets observations (a metric or distance function) as well as a linkage criterion which determines the distance between clusters. Thus, the linkage criterion is a function of the chosen metric. For example, let a and b be two vectors and let A and B represent two sets of observations (clusters). One might choose the squared Euclidean distance metric, which is $d(a,b) = \left\|a-b\right\|^2 = \sum_i (a_i - b_i)^2$. Simultaneously, the single-linkage criterion may be chosen, which is defined by $\min\{d(a,b) : a \in A, b \in B\}$. These criteria govern how the hierarchical clustering scheme will be built. In some sense, there is much flexibility in this method since any valid measure of distance and linkage may be used.

1.2.1 Convexifying Hierarchical Clustering

Recall that the optimization problem for hierarchical clustering (1.7) is a difficult combinatorial optimization problem. After noting this, Hocking et. al. (2011) proposed a convex relaxation of (1.7) defined by

$$\begin{aligned} \min_{M \in \mathbb{R}^{n \times d}} \quad & \frac{1}{2}\left\|X - M\right\|_F^2 \\ \text{subject to} \quad & \sum_{i<j} \|M_i - M_j\|_q \leq t, \end{aligned} \tag{1.8}$$

where $\|\cdot\|_q$, $q \geq 1$, is the L_q norm, which is able to shrink its argument to exactly zero (we omit the weights here for simplicity. The issue of weights is discussed in more detail in Section 2.1). The parametrization in terms of t is inconvenient, so the equivalent Lagrangian formulation is often used:

$$\min_{M \in \mathbb{R}^{n \times d}} \quad \frac{1}{2}\big\|X - M\big\|_F^2 + \lambda \sum_{i<j} \|M_i - M_j\|_q \tag{1.9}$$

One can imagine how varying the values of t and λ in (1.8) and (1.9), respectively, controls the amount of shrinkage, which is equivalent to controlling the number of clusters. Notice that the move from (1.7) to (1.8) *convexifies* the optimization problem, so that (1.8) is a *convex relaxation* of (1.7). Speaking loosely, we went from a kind of L_0 "norm" to the L_q norm, which is convex for $q \geq 1$.

Chapter 2

Overview of Solution Path Clustering

2.1 Convex Clustering

"Traditional clustering methods such as K-means clustering, hierarchical clustering, and Gaussian mixture models take a greedy approach and suffer from instabilities due to their nonconvex optimization formulations" (Wang, et. al. 2018). The nonconvex formulations of these classic clustering methods lead to their common weaknesses of local sub-optima and initialization problems. One might naturally consider a convex formulation of the clustering problem in order to solve those issues (for example, see the ends of Sections 1.1 and 1.2). This is a primary motivation in developing *convex clustering*. Alternatively, one can arrive at the convex clustering in (2.1) by assuming that each $y_i \sim N(m_i, I)$ and then adding a regularization (penalty) term. Section 2.4 contains developments and refer-

ences in convex clustering.

Convex clustering casts the clustering objective as a convex optimization problem and can thus admit a global solution. The convex clustering objective function can be written as

$$Q(M, \lambda) = \frac{1}{2} \sum_{i=1}^{n} (y_i - m_i)^T (y_i - m_i) + \lambda \sum_{i<j} \left\| m_i - m_j \right\|_q \tag{2.1}$$

where $\left\| \cdot \right\|_q$ is the L_q norm. Note that q must be greater than or equal to 1 for objective function (2.1) to remain convex, although it is possible to extend it to the non-convex case $0 < q < 1$ (Wang, et. al. 2016). The cases $q \in \{1, 2, \infty\}$ are the most commonly considered, with $q = 2$ being perhaps the most sensible choice for clustering. If $q = 2$ (in fact, any $q > 1$), the $m_i - m_j$ differences will be shrunk to the zero vector which is needed to group them to the same cluster. If $q = 1$ is used, only specific dimensions will be shrunk to zero and two observations will not be clustered together until all of their component means are simultaneously zero. The situation is analogous when considering the group-LASSO in which groups of variables are shrunk to zero (Yuan and Lin, 2006).

The m_i represents simultaneously the cluster center and membership of y_i. Due to the sparsity inducing penalty, some of the $|m_i - m_j|$ will shrink to exactly zero which partitions the data into clusters. The common value shared between any merged pair of m_i and m_j is the estimated cluster center. For a fixed λ, let $\hat{K}$ be the resulting number of clusters and let $\{\hat{\alpha}_1, \hat{\alpha}_2, \ldots, \hat{\alpha}_{\hat{K}}\}$ be the unique values of $\{\hat{m}_1, \hat{m}_2, \ldots, \hat{m}_n\}$. Let $\hat{G}_j = \{i : \hat{m}_i = \hat{\alpha}_j\}$ for $1 \leq j \leq \hat{K}$. Then $\{\hat{G}_1, \hat{G}_2, \ldots, \hat{G}_{\hat{K}}\}$ forms a partition of $\{1, 2, \ldots, n\}$.

The tuning parameter $\lambda > 0$ controls the amount of shrinkage (penalization, regularization) of the pairwise differences of the m_i and thus controls the number of clusters.

The penalty term shrinks some of the $m_i - m_j$ to exactly zero in the same spirit that LASSO regression performs variable selection by shrinking coefficients to exactly zero (Tibshirani 1996). When $\lambda = 0$, the objective function (2.1) is minimized when $m_i = y_i$ for each i and each observation is its own cluster for a total of n clusters. When $\lambda \to \infty$, it is minimized when all of the m_i are equal to a common value (the mean) as all observations are merged into one cluster. Moderate values of λ yield cluster solutions with a number of clusters between 1 and n. A *solution path* of clustering results is obtained by finding the minimizers $\hat{m}(\lambda)$ along a grid of λ values. Thus, convex clustering can be a powerful exploratory device as it outputs a tree-like structure similar to hierarchical clustering.

In some sense, it seems that convex clustering is only complicating matters since it introduces as many parameters as there are observations, and then relies on regularization to prevent over-fitting from the many parameters it just introduced. A motivating factor was to move the clustering problem to a convex problem, but are there further benefits to such a formulation? Recall that some clustering methods, such as K-means and mixture model clustering, require specifying the number of components as an input argument for the clustering algorithm. Instead of specifying the number of components, convex clustering requires specifying the tuning parameter λ. The choice of λ effectively controls the number of clusters, but it is more accurate to say that it *dynamically* controls the trade-off between model fit and the number of clusters. It is dynamic because convex clustering has the ability to adapt the number of clusters to a changing dataset. For example, consider sequential data that changes over time. A static method such as K-means will fit the same number of clusters throughout the sequential data process. On the other hand, convex clustering

can adapt the number of clusters even with the same value of λ. Of course, it is possible to apply order selection methods, but this would need to be done each time clustering is performed on the sequential data. However, it is also unclear if a fixed λ value will perform best. See (Lindsten, Ohlsson, and Ljung, 2011a) for an example and discussion.

A further benefit is that the solution path is unique and depends continuously on λ (see Proposition 2.1 in Chi and Lange, 2015). This is reminiscent to the continuous variable selection properties of LASSO regression. That the solutions $\hat{m}(\lambda)$ are unique and global solutions was an original motivating factor in forming a strictly convex objective function. The continuity property justifies using *warm starts* (discussed in Section 2.3) in constructing the solution path and the computational time saved is appreciable.

The benefits gained from convexification come at a cost. The use of a convex penalty in the objective function can seriously bias the results and ruin the search for clusters. For example, consider two points, say y_i and y_j, that are far apart from one another. Moderate choices of λ will likely not cluster them together, but a convex penalty will over-penalize the difference $m_i - m_j$ and cause their estimates to be biased, sometimes extremely so. Lindsten, Ohlsson, and Ljung (2011a) suggest a post-processing step as a possible remedy. After finding the minimizer of objective function (2.1), we have the estimated cluster membership of each observation because we know which pairs $\hat{m}_i - \hat{m}_j$ are shrunk to exactly zero. This creates the cluster partitions G_j, $j = 1, 2, \ldots, \hat{K}$ for estimated number of clusters $\hat{K}$ (see description of the G_j notation in Section 1.1). Then simply compute the sample means of the y_i within each cluster:

$$\hat{\alpha}_j = \frac{1}{\text{card } G_j} \sum_{i \in G_j} y_i, \quad j = 1, 2, \ldots, \hat{K}$$

Here, the $\hat{\alpha}_j$ represent the cluster means. If desired, we may then make the update assignment

$$\hat{m}_i^* = \sum_{j=1}^{\hat{K}} \hat{\alpha}_j I(y_i \in G_j)$$

where $I(\cdot)$ is the indicator function. This post-processing step is relevant only if we care about the actual values of the m_i and we want to remove the bias. If we only care about the clustering partition itself, this step may be skipped since we only need to know which pairs $\hat{m}_i - \hat{m}_j$ are shrunk to zero and not their actual values. Figure 2.1 illustrates the amount of bias that can be present when this post-processing step is not used. Note that this strategy does not address the bias that occurs *within* the estimation procedure itself before the solution is found, which may be detrimental in the search for clusters (Ma and Huang, 2017).

Lindsten, Ohlsson, and Ljung (2011a) suggest multiplying the penalty term in (2.1) by weights to prevent over-penalizing points that are already far apart. The objective function then becomes

$$Q(M, \lambda) = \frac{1}{2} \sum_{i=1}^{n} (y_i - m_i)^T (y_i - m_i) + \lambda \sum_{i<j} w_{ij} \left\| m_i - m_j \right\|_q \tag{2.2}$$

where the weights $w_{ij} \geq 0$ are pre-specified. Using weights will, in general, slow the merging of clusters. In fact, objective function (2.2) is how Hocking, et. al. (2011) presented the convex clustering problem. In their article, they suggested a Gaussian weight

$$w_{ij} = \exp[-c(y_i - y_j)^T (y_i - y_j)] \tag{2.3}$$

where $c > 0$. These weights decay as the distance between y_i and y_j increases, counteracting the over-penalization. See Figure 2.2 for solution paths employing Gaussian weights.

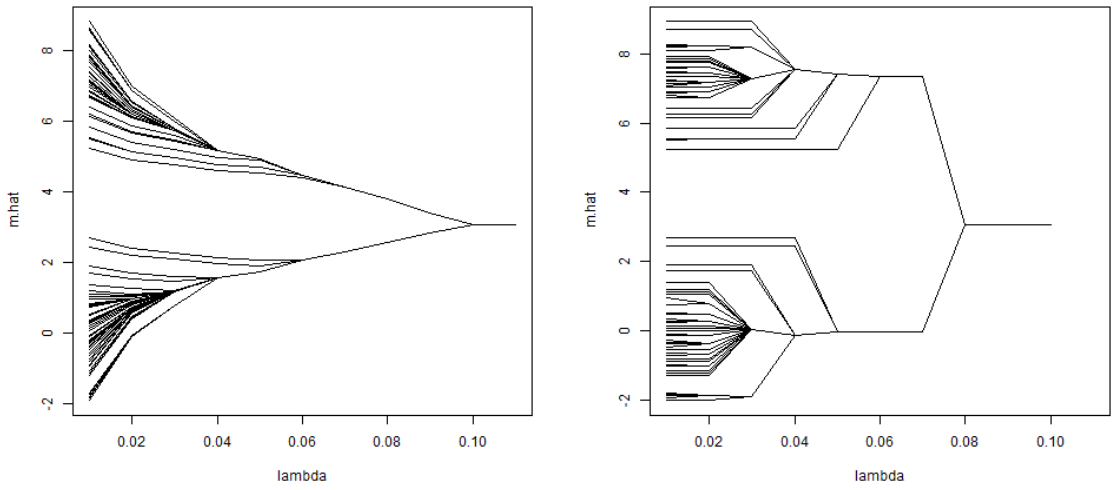
Lindsten, Ohlsson, and Ljung (2011a) used a k-nearest-neighbor weight

$$w_{ij} = \begin{cases} 1 & \text{if } y_i \in k\text{NN}(y_j) \text{ or } y_j \in k\text{NN}(y_i) \\ 0 & \text{otherwise,} \end{cases} \tag{2.4}$$

where $k\text{NN}(y_i)$ is the set of the k nearest neighbors of y_i. See Figures 2.3b and 2.3c for solution paths employing kNN weights. Chi and Lange (2015) suggest using a combination of the weights (2.3) and (2.4). This yields

$$w_{ij} = \kappa_{ij} \exp[-c(y_i - y_j)^T (y_i - y_j)] \tag{2.5}$$

where κ_{ij} is defined as in (2.4). In the same article, they discuss how this choice of weights "improves both computational efficiency and clustering quality," and how the combination



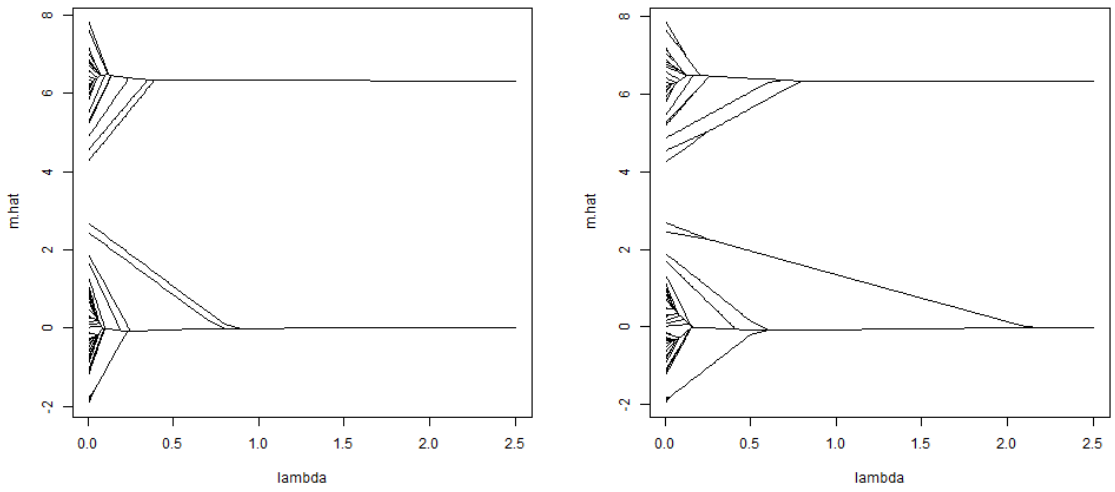
(a) Solution path without post-processing step.

(b) Solution path with post-processing step.

Figure 2.1: Solution path plots of $\hat{m}(\lambda)$ against λ. There are 100 total observations generated from $\frac{1}{2}N(\mu = 0, \sigma^2 = 1) + \frac{1}{2}N(\mu = 7, \sigma^2 = 1)$.

of (2.3) and (2.4) "increases the sensitivity of the clustering path to the local density of the data." See Figure 2.3d for a solution path using Gaussian-kNN weights.

Note that the choice of weights can dramatically affect the solution path (see Figures 2.2 and 2.3) and there is no general rule for how to choose the weights (Ma and Huang 2017). An exception is Chen, et. al. (2015) where the authors discuss some heuristics for weight choices in biology contexts. Though the consideration of weights adds flexibility in the behavior of the solution path, the user is burdened with fine-tuning the exact weights to be used, such as the c value in the Gaussian weights or the number of nearest neighbors.

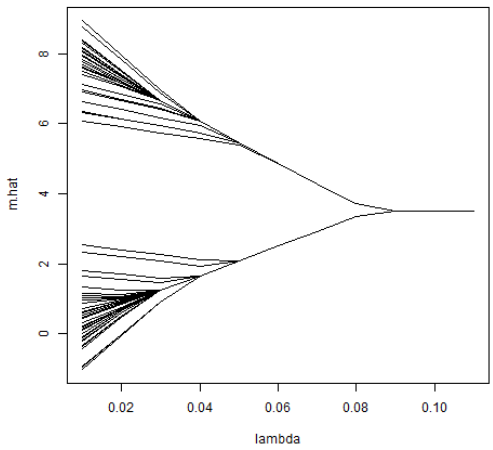


(a) Solution path with Gaussian weights defined by equation (2.3) with $c = 0.75$.

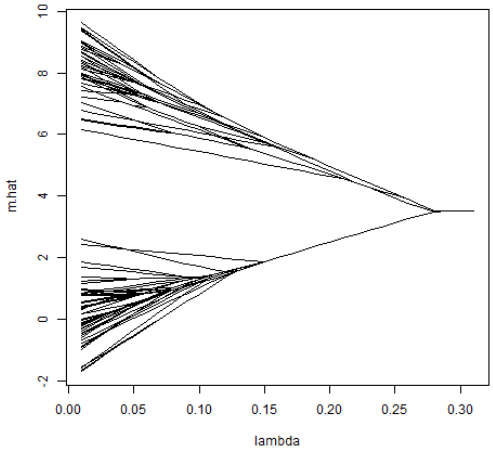
(b) Solution path with Gaussian weights defined by equation (2.3) with $c = 1.5$.

Figure 2.2: Solution path plots of $\hat{m}(\lambda)$ against λ. There are 100 total observations generated from $\frac{1}{2}N(\mu = 0, \sigma^2 = 1) + \frac{1}{2}N(\mu = 6, \sigma^2 = 1)$. Note that the post-processing effect was not used in these solution paths.

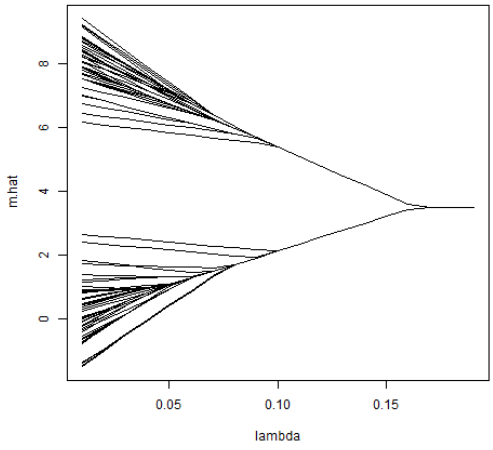
A final thing to note is that the solution path might not be agglomerative; that is, after a pair $(\hat{m}_i, \hat{m}_j)$ has been merged, it is possible that they will be un-merged later in the solution path. Hocking, et. al. (2011) provide an example in which such a split occurs. However, they also provide a theoretical guarantee (see their Theorem 1) that no splits will occur in the special case of L_1 penalty and uniform weights $w_{ij} = 1$. Even outside this special case, splits do not seem to occur very often in practice, though technically they need to be considered in order to ensure that the global solution is indeed found (Chi and Lange 2015).



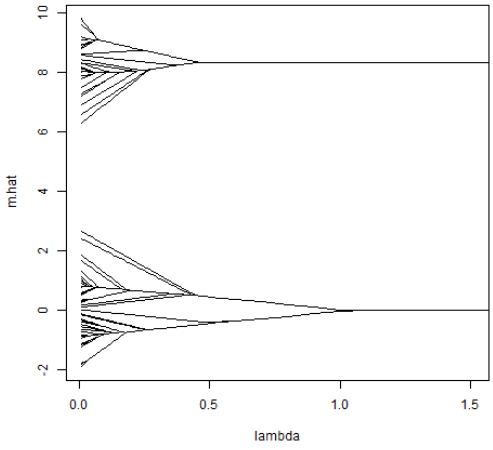
(a) Solution path with uniform weights $w_{ij} = 1$ for all i and j.



(b) Solution path with kNN weights defined by equation (2.4) using $k = 15$ nearest neighbors.



(c) Solution path with kNN weights defined by equation (2.4) using $k = 25$ nearest neighbors.



(d) Solution path with Gaussian-kNN weights (2.5) using $c = 0.5$ and $k = 15$ nearest neighbors.

Figure 2.3: Solution path plots of $\hat{m}(\lambda)$ against λ. There are 100 total observations generated from $\frac{1}{2}N(\mu = 0, \sigma^2 = 1) + \frac{1}{2}N(\mu = 8, \sigma^2 = 1)$. Note that the post-processing effect was not used in these solution paths.

2.2 Forming Clusters with Concave Penalties

In this section, we consider univariate data, although it is straightforward to extend the derivations here to multivariate data (see Section 4.1). Consider the objective function

$$Q(m, \lambda) = \frac{1}{2} \sum_{i=1}^{n} (y_i - m_i)^2 + \sum_{i<j} p(|m_i - m_j|, \lambda) \tag{2.6}$$

where $|\cdot|$ is the L_1 norm and $p(\cdot, \lambda)$ is a penalty function. For example, if $p(|x|, \lambda) = \lambda|x|$, then we recover the convex clustering objective function (2.1). A good penalty should result in estimators that exhibit the properties of unbiasedness, sparsity, and continuity (Fan and Li, 2001). Motivated by these criteria, addressing especially the biasedness problems of the L_1 penalty, Ma and Huang (2017) proposed using the Smoothly Clipped Absolute Deviations penalty (SCAD) from Fan and Li (2001), and the Minimax Concave Penalty (MCP) from Zhang (2010), both of which are concave penalties. Similarly motivated, Marchetti and Zhou (2014) also used the MCP. We focus on MCP in this work, although SCAD provides similar results (Ma and Huang, 2017).

Let $t \geq 0$. The MCP can be written as

$$p_\omega(t, \lambda) = \lambda \int_0^t \left(1 - \frac{x}{\omega\lambda}\right)_+ dx, \omega > 1, \tag{2.7}$$

where $(x)_+ = \max(0, x)$ and $\omega > 0$ controls the level of concavity. It is helpful to also write the MCP as

$$p_\omega(t, \lambda) = \begin{cases} \lambda t - \dfrac{t^2}{2\omega} & \text{if } t \leq \omega\lambda \\ \dfrac{\lambda^2 \omega}{2} & \text{if } t > \omega\lambda. \end{cases}$$

Following Zhang (2010), we treat ω as a fixed constant, although it is possibly to vary as in Marchetti and Zhou (2014). In fact, when $\omega \to \infty$, the MCP converges to the L_1 penalty,

and when $\omega \to 0+$, the MCP converges to the L_0 penalty (Marchetti and Zhou, 2014). See Figure 2.4 for a plot of the MCP for different choices of ω.

Note that the MCP becomes constant after a certain threshold. This feature prevents unnecessary penalization when the true distance between clusters is large. Contrast this to the L_1 penalty which continues to grow. This causes unnecessary shrinkage and

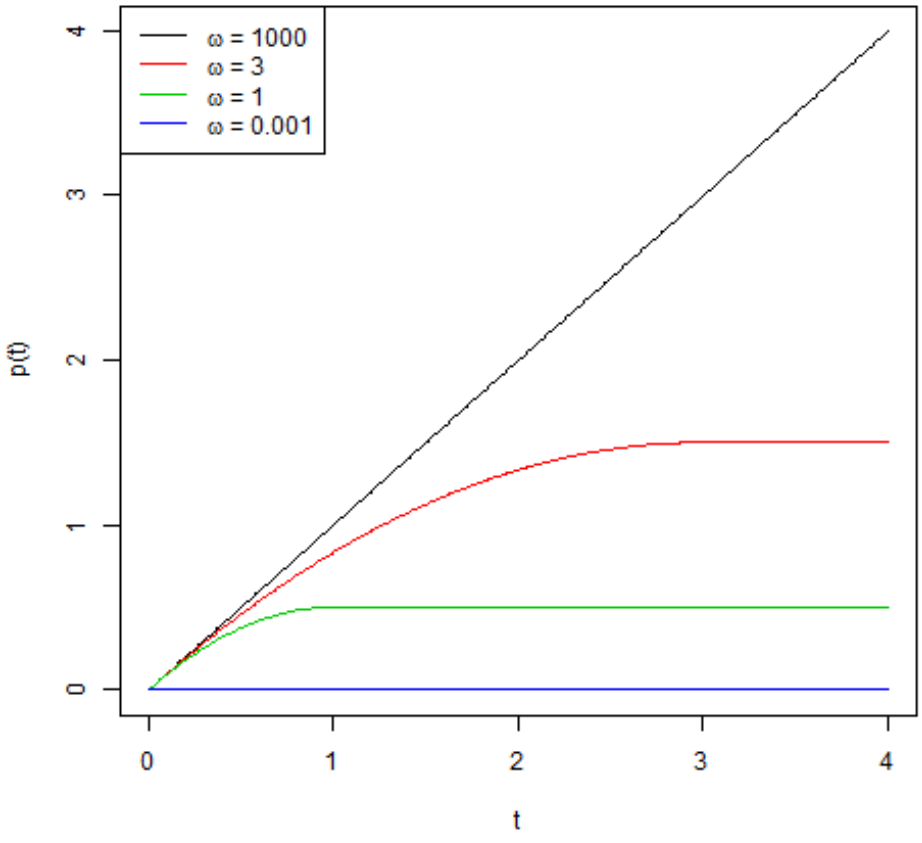


Figure 2.4: Plot of MCP penalty with $\lambda = 1$ and different values of the concavity parameter ω.

results in biased estimates – so much so that it can ruin the search for subgroups. The MCP retains the sparsity and continuity properties while remaining unbiased. Comparing Figure 2.5 and 2.6, we see that the MCP penalty performs better in the search for subgoups and exhibits a reasonable range of λ values where the true cluster means are discovered. The L_1 penalty, depending on the weight choices, can produce either many subgroups or

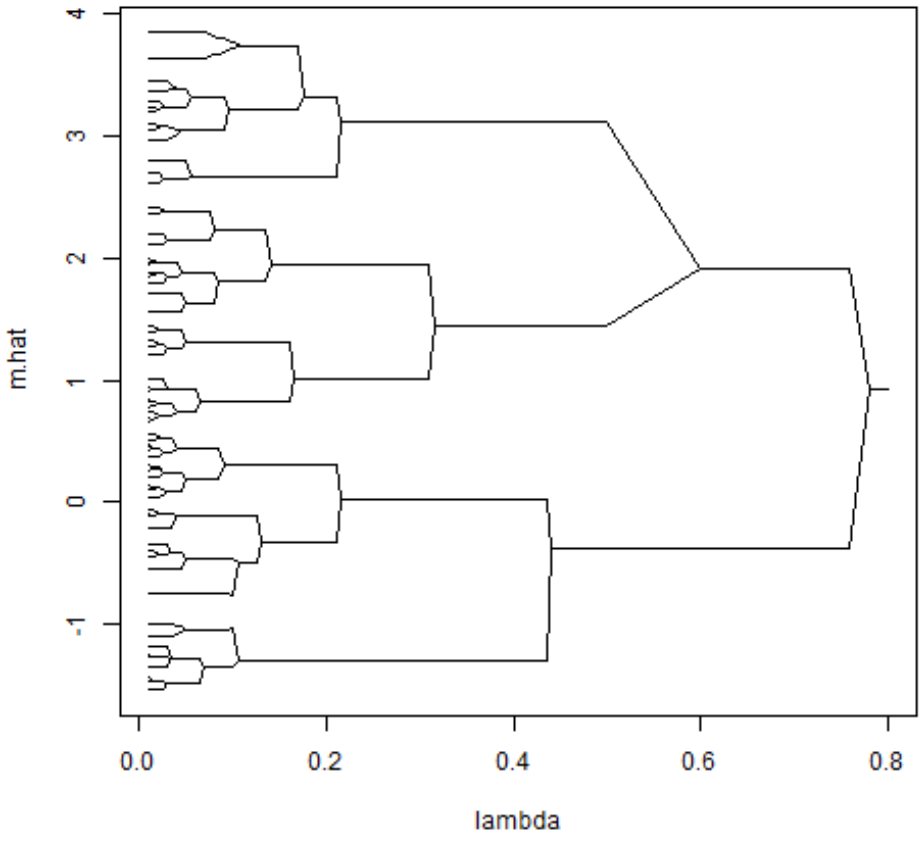
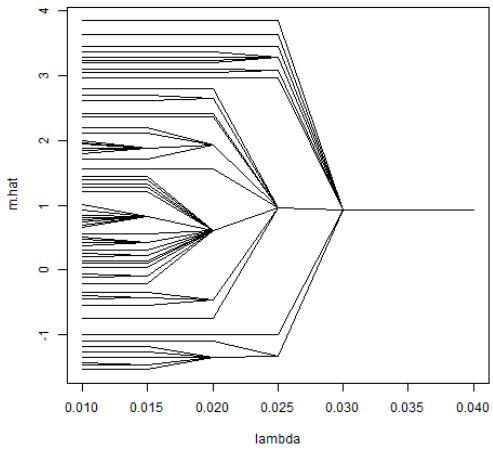


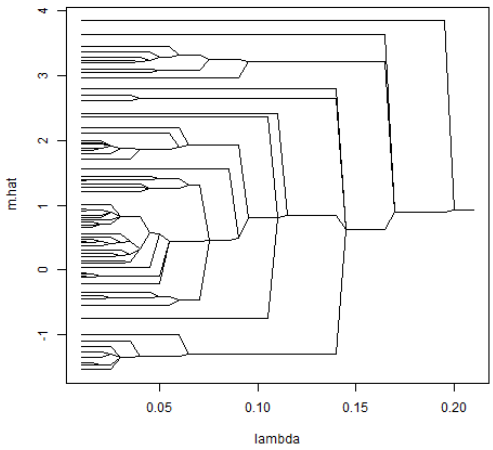
Figure 2.5: Solution path plot with MCP penalty of $\hat{m}(\lambda)$ against λ. There are 100 total observations generated from $\frac{1}{2}N(\mu = 0, \sigma^2 = 1) + \frac{1}{2}N(\mu = 2, \sigma^2 = 1)$. Note the same dataset was used here as in Figure 2.6

no subgroups, moving extremely fast from the former to the latter for small increases in λ.

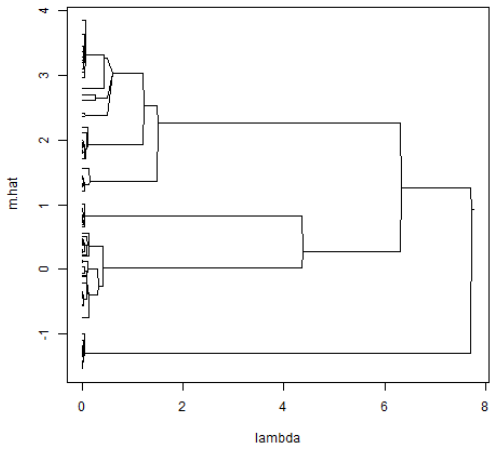
The move from convex penalties to concave penalties may seem a bit strange. A major reason for using convex penalties was to guarantee that the global optimum can be found, and we no longer have this guarantee when a concave penalty is used. A major drawback of convex penalties, however, is the biasedness of the resulting estimates, especially when the distance between cluster centers is large. The bias can be partly mitigated through a careful choice of weights, but an important question remains: What is the best choice of weights for my dataset? In Figure 2.6, a correct choice of weights will lead to a solution path that more closely recovers the true cluster means, but a poor choice of weights will not. Concave penalties, on the other hand, eliminate any need to choose weights. Figure2.5, which is a solution path on the same dataset as Figure 2.6, shows that the MCP penalty can cleanly recover the true cluster means, and it does so within a healthy range of λ values. Furthermore, when using the MCP penalty, there is no need for a post-processing step of computing sample means based on cluster membership.



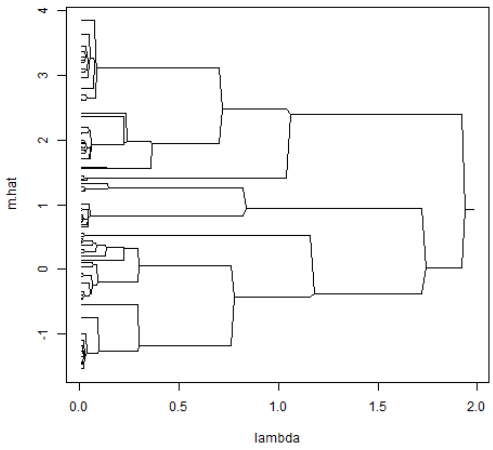
(a) Solution path with uniform weights $w_{ij} = 1$ for all i and j.



(b) Solution path with Gaussian weights defined by equation (2.3) using $c = 0.5$ nearest neighbors.



(c) Solution path with Gaussian-kNN weights (2.5) using $c = 0.5$ and $k = 10$ nearest neighbors.



(d) Solution path with Gaussian-kNN weights (2.5) using $c = 0.5$ and $k = 15$ nearest neighbors.

Figure 2.6: Solution path plots of $\hat{m}(\lambda)$ against λ using L_1 penalty with weights and the post-processing step. There are 100 total observations generated from $\frac{1}{2}N(\mu = 0, \sigma^2 = 1) + \frac{1}{2}N(\mu = 2, \sigma^2 = 1)$. Note that the same dataset was used here as in Figure 2.5

2.3 ADMM Algorithm

The Alternating Direction Method of Multipliers (ADMM) algorithm is a powerful and efficient algorithm that is well-suited for optimizing convex objective functions. An extremely thorough introduction and review of the ADMM algorithm is provided by Boyd, et. al. (2011). The key reference for how the ADMM algorithm is applied to convex clustering is Chi and Lange (2015). It may also be useful to reference Parikh and Boyd (2013) since the ADMM steps can be viewed through the theory of proximal mappings.

The ADMM is well-suited for minimizing convex functions and so it fits perfectly for the convex clustering criterion (2.1). Ma and Huang (2017) also proved the ADMM converges when concave penalties are used. Here, we will not develop the general ADMM algorithm and then apply it to our case. Instead, we will arrive at the ADMM algorithm directly by considering the following problem:

$$\min_{m \in \mathbb{R}^n} \frac{1}{2} \sum_{i=1}^{n} (y_i - m_i)^2 + \sum_{i<j} p_\omega\big(|m_i - m_j|, \lambda\big) \tag{2.8}$$

We will thus recover the ADMM algorithm from the "ground up," so to speak, following a similar development in Ma and Huang (2017). We consider here only univariate data. For the multivariate version, see Section 4.1.

Directly minimizing the objective function in (2.8) is difficult because the penalty term is not separable in the m_i's. To overcome this, we use the variable splitting technique (Chi and Lange, 2015; Boyd, et. al. 2011). Reparametrize the objective in (2.8) by

introducing a new set of parameters $\eta_{ij} = m_i - m_j$ to obtain

$$\begin{aligned} S(m, \eta) &= \frac{1}{2}\sum_{i=1}^{n}(y_i - m_i)^2 + \sum_{i<j} p_\omega(|\eta_{ij}|, \lambda) \\ &\text{subject to } m_i - m_j - \eta_{ij} = 0 \end{aligned} \tag{2.9}$$

where $\eta = \{\eta_{ij}, i < j\}$. Now, encode the constraint with the augmented Lagrangian:

$$\begin{aligned} L_\beta(m, \eta, \nu) = \frac{1}{2}\sum_{i=1}^{n}(y_i - m_i)^2 + \sum_{i<j} p_\omega(|\eta_{ij}|, \lambda) \\ + \sum_{i<j}\nu_{ij}(m_i - m_j - \eta_{ij}) + \frac{\beta}{2}\sum_{i<j}(m_i - m_j - \eta_{ij})^2 \end{aligned} \tag{2.10}$$

where $\beta > 0$ is a penalty parameter and $\nu = \{\nu_{ij}, i < j\}$ is a $\binom{n}{2} \times 1$ vector of dual variables. We have now transformed the difficult problem (2.8) to an unconstrained objective function. The objective function (2.10) is the exact form to which ADMM updates are applied. Before deriving the updates, it will be convenient to rewrite (2.10) into matrix form. Let $\Delta = [(e_i - e_j), i < j]^T$ be the $\binom{n}{2} \times n$ matrix composed of $n \times 1$ vectors e_i, in which the i-th element is 1 and the remaining elements are 0. Pre-multiplying this Δ matrix to the m vector forms the $\binom{n}{2}$ vector composed of all pairwise $m_i - m_j$, $i < j$. Then (2.10) can be expressed as

$$\begin{aligned} L_\beta(m, \eta, \nu) = \frac{1}{2}\sum_{i=1}^{n}(y_i - m_i)^2 + \sum_{i<j} p_\omega(|\eta_{ij}|, \lambda) \\ + \nu^T(\Delta m - \eta) + \frac{\beta}{2}\|\Delta m - \eta\|_2^2 \end{aligned} \tag{2.11}$$

For the t-th iteration, the ADMM updates are

$$m^{(t+1)} := \underset{m}{\operatorname{argmin}}\, L_\beta(m, \eta^{(t)}, \nu^{(t)}) \qquad \text{(Step 1)}$$

$$\eta^{(t+1)} := \underset{\eta}{\operatorname{argmin}}\, L_\beta(m^{(t+1)}, \eta, \nu^{(t)}) \qquad \text{(Step 2)}$$

$$\nu^{(t+1)} = \nu^{(t)} + \beta(\Delta m^{(t+1)} - \eta^{(t+1)}) \qquad \text{(Step 3)}$$

In what follows, we use the "hat" to denote the update and it is understood that any update is calculated using the current estimate of the other parameters.

Step 1 in the ADMM algorithm is to minimize (2.11) with respect to m. We can find the exact update in the standard way by setting the gradient equal to zero and then solving, which yields

$$\hat{m} = \big(I + \beta\Delta^T\Delta\big)^{-1}\big[y + \beta\Delta^T(\eta - \beta^{-1}\nu)\big] \tag{2.12}$$

Note that $I + \beta\Delta^T\Delta = (1 + n\beta)I - \beta\mathbf{1}\mathbf{1}^T$ where $\mathbf{1}$ is a $n \times 1$ vector of ones. The right hand side is a convenient representation to which we can apply the Sherman-Morrison formula for a fast calculation of the inverse:

$$\big[(1 + n\beta)I - \beta\mathbf{1}\mathbf{1}^T\big]^{-1} = \frac{1}{1 + n\beta}\big(I + \beta\mathbf{1}\mathbf{1}^T\big)$$

We can thus avoid any numerical routine for calculating the inverse of a matrix.

Step 2 is to minimize (2.11) with respect to η. Note that the term involving η is simply a sum of each of the η_{ij}'s, and so we only need to derive the update for η_{ij} which is then applied to each of them. It can be shown under certain conditions that L_β is convex with respect to each η_{ij} when all other function arguments are held fixed (note, however, that L_β is not a convex function when a concave penalty is used). Minimizing L_β with respect to η_{ij} is equivalent to minimizing

$$p(|\eta_{ij}|) + \frac{\beta}{2}(\delta_{ij} - \eta_{ij})^2 \tag{2.13}$$

where $\delta_{ij} = m_i - m_j + \beta^{-1}\nu_{ij}$. Consider first the L_1 penalty where $p(|\eta_{ij}|) = \lambda|\eta_{ij}|$. Since the absolute value function is not differentiable everywhere, we must derive the update by

using sub-gradients instead of gradients (Boyd and Vandenberghe, 2008). Let the operator ∂ denote the sub-gradient with respect to η_{ij}. The optimiality condition is

$$\begin{aligned} & 0 \in \partial\Big(\lambda|\eta_{ij}| + \frac{\beta}{2}(\eta_{ij} - \delta_{ij})^2\Big) \\ \implies & 0 \in \lambda\partial|\eta_{ij}| + \beta\eta_{ij} - \beta\delta_{ij} \end{aligned}$$

The two cases to consider are if $\eta_{ij} = 0$ and if $\eta_{ij} \neq 0$.

If $\eta_{ij} = 0$, then the optimality condition becomes

$$\begin{aligned} & 0 \in -\beta\delta_{ij} + \lambda[-1, 1] \\ \iff & \delta_{ij} \in [-\frac{\lambda}{\beta}, \frac{\lambda}{\beta}] \\ \iff & |\delta_{ij}| \leq \frac{\lambda}{\beta} \end{aligned}$$

If $\eta_{ij} \neq 0$, then the optimiality condition becomes

$$\begin{aligned} & 0 = \lambda \text{ sign}(\eta_{ij}) + \beta\eta_{ij} - \beta\delta_{ij} \\ \implies & \hat{\eta}_{ij} = \delta_{ij} - \frac{\lambda}{\beta} \text{ sign}(\hat{\eta}_{ij}) \end{aligned}$$

Note that if $\hat{\eta}_{ij} < 0$, it would mean that $\delta_{ij} + \frac{\lambda}{\beta} < 0 \iff \delta_{ij} < -\frac{\lambda}{\beta}$.

Similarly, if $\hat{\eta}_{ij} > 0$, it would mean that $\delta_{ij} - \frac{\lambda}{\beta} > 0 \iff \delta_{ij} > \frac{\lambda}{\beta}$.

Putting these together yields the conclusions that $|\delta_{ij}| > \frac{\lambda}{\beta}$, and that $\text{sign}(\hat{\eta}_{ij}) = \text{sign}(\delta_{ij})$.

Thus, we have

$$\begin{aligned} \hat{\eta}_{ij} &= \begin{cases} 0 & \text{if} \quad |\delta_{ij}| \leq \frac{\lambda}{\beta} \\ \delta_{ij} - \frac{\lambda}{\beta} \text{ sign}(\delta_{ij}) & \text{if} \quad |\delta_{ij}| > \frac{\lambda}{\beta} \end{cases} \\ &= \text{sign}(\delta_{ij})\Big(|\delta_{ij}| - \frac{\lambda}{\beta}\Big)_+ \\ &:= \text{ST}\Big(\delta_{ij}, \frac{\lambda}{\beta}\Big) \end{aligned}$$

a soft thresholding update.

The update for the MCP penalty is derived similarly. For $\omega > \beta^{-1}$, the update is

$$\hat{\eta}_{ij} = \begin{cases} \dfrac{\mathrm{ST}\big(\delta_{ij}, \lambda/\beta\big)}{1 - 1/(\omega\beta)} & \text{if } |\delta_{ij}| \leq \omega\lambda \\ \delta_{ij} & \text{if } |\delta_{ij}| > \omega\lambda \end{cases} \tag{2.14}$$

Note that these penalties can lead to updates that set $\hat{\eta}_{ij}$ to exactly zero for sets of δ_{ij} as controlled by λ (both ω and β are fixed). Mathematically, this comes from the fact that these penalties are not differentiable at zero and subgradients must be used.

The Step 3 update can be derived by the method of steepest ascent for maximizing the dual. Since the update depends on the previous update, it can be seen as a sort of running sum of errors against the constraint.

Steps 1, 2, and 3 are cycled through and repeated until some convergence criterion is met. Convergence is judged by a threshold on the dual and the primal residuals. For iteration t, the primal residual is $r^{(t+1)} = \Delta m^{(t)} - \eta^{(t)}$, and the dual residual is $s^{(t+1)} = \beta\Delta^T\big(\eta^{(t+1)} - \eta^{(t)}\big)$. If they are small (say, below some small $\epsilon > 0$), then the algorithm is terminated. See Boyd, et. al. (2011) for some guidance on the convergence criterion.

Consider the initial values

$$\begin{aligned} m^{(0)} &= y \\ \eta^{(0)} &= \Delta m^{(0)} \\ \nu^{(0)} &= 0 \end{aligned} \tag{2.15}$$

In constructing the solution path, we use the *warm start* strategy which can be described as follows. Let $\lambda_1 < \lambda_2 < \cdots < \lambda_{\max}$ be a sequence of λ values at which we will compute the solutions $\hat{m}(\lambda)$. The smallest λ value, λ_1, is initialized as defined by (2.15). The next

λ value in the solution path sequence, λ_2, is then initialized by

$$m^{(0)} = \hat{m}(\lambda_1)$$

$$\eta^{(0)} = \Delta\hat{m}(\lambda_1)$$

$$\nu^{(0)} = 0$$

and so on and so forth for the following λ values. The solution path construction can be stopped as soon as the λ value is large enough such that all m_i are merged into a single cluster.

Upon convergence, and if λ is suitably large enough, some of the $\hat{\eta}_{ij}$ will be exactly zero. Clusters are formed by putting the i and j into the same cluster if $\hat{\eta}_{ij} = 0$. Note that even though $\hat{\eta}_{ij}$ will be exactly zero, it is possible that $\hat{m}_i - \hat{m}_j \neq 0$, although it will be extremely close to zero. We can simply estimate the k-th cluster center by

$$\hat{\alpha}_k = \frac{1}{n_k} \sum_{i \in \hat{G}_k} \hat{m}_i$$

where $\hat{G}_k$ is the estimated cluster membership (based on the $\hat{\eta}_{ij}$) and n_k is the number of elements in $\hat{G}_k$.

When a convex penalty is used, there is a unique minimum point for each value of λ (Chi and Lange, 2015). When a concave penalty is used, Ma and Huang (2017) showed that the ADMM algorithm will still converge, albeit to a local minimum. Using the *warm start* strategy described above, not only is the convergence quicker, but the solution quality is generally good. Another way to improve the solution quality is to implement a *merging step* (described below). However, even with the warm start strategy and merging step, it is possible that good solutions are not found. See the simulation studies in this

dissertation, especially the discussion on viability rates. In particular, if K is the true number of components, it is possible that the solution path calculated for a given λ grid does not contain a K-component solution.

With a concave penalty, one issue that we have experienced is that the ADMM algorithm will often get stuck in local sub-optimal minima that can be easily improved. In particular, the resulting cluster partition can sometimes exhibit some clusters that are comprised of very few observations, such as three or below. To combat this issue, we use a *merging step* that can be described as follows. Let $n_{\min}$ be the smallest cluster size, and let $\bar{n}$ be the average cluster size. If upon convergence, we find that

$$\frac{n_{\min}}{\bar{n}} < \tau \tag{2.16}$$

where τ is a proportion (for example, $\tau = 0.20$), then the smallest cluster is merged to the cluster closest to it in Euclidean distance. To merge, we simply assign the m_i in the smallest cluster to the cluster value that it is joining. Re-label and repeat this merging step until (2.16) is no longer satisfied. Then, run the ADMM algorithm again using these new $\hat{m}^*$ as the initial value. Usually, it will only run for a few iterations. To justify this merging step, we only keep the merged solution if it computes an objective function value smaller than that of the solution before the merge(s). Note that a different criterion or another strategy may be used. The goal here is only to find a better local minimum. We have observed that most of the time, the merged solution is better than the solution before the merge.

2.4 Literature Review

In this section, we provide a quick overview of the publications pertaining to solution path clustering. It is not exhaustive, though we have attempted to mention the important publications and their contributions.

To our knowledge, convex clustering as presented in the form (2.1) was first proposed by Pelckmans, et. al. (2005). They also presented an efficient quadratic convex programming method for computing the estimates when the L_1 penalty is used. Lindsten, Ohlsson, and Ljung (2011a) then presented convex clustering under the name "sum-of-norms regularization." They suggest that the L_2 norm (or the L_q norm with $q > 1$) to be the most appropriate penalty choice and mentioned the analogous situation between LASSO (Tibshirani, 1996) and Group-LASSO (Yuan and Lin, 2006). Some choices of weights are discussed. They used a off-the-shelf convex optimization package, though they provide their code for easy implementation. The same authors also published a companion technical report (Lindsten, Ohlsson, and Ljung, 2011b) that develops the convex clustering objective as a convex relaxation of K-means. Similarly, Hocking, et. al. (2011) motivated convex clustering as a convex relaxation of hierarchical clustering. They derive dedicated algorithms for the cases of L_1, L_2 and L_∞ penalty cases. They also proved a theorem that the solution path with the L_1 penalty and uniform weights contains no splits, and provided an example that splits can occur when the L_2 penalty is used.

Pan, Shen, and Liu (2013) appear to be the first to use a concave penalty to fuse cluster centers. In particular, they use the truncated LASSO penalty (Shen, Pan, and Zhu, 2012). They emphasize that solution path clustering can be viewed as a penalized regression

so that machinery developed in regression contexts may be borrowed and applied to the cluster analysis. They call their method Penalized Regression-based Clustering (PRclust). Interestingly, they use a splitting technique to compute the estimates which is quite close to the ADMM algorithm. They propose to select the number of clusters using generalized cross-validation (Golub, Heath, and Wahba, 1979) based on generalized degrees of freedom (Ye, 1998). Later, Wu, et. al. (2016) extended the PRclust method to use the ADMM algorithm with a difference in convex programming step and demonstrated that it is much more efficient. They also proved a clustering consistency result of the PRclust method under certain conditions. They proposed to select the number of clusters based on cross-validation and a stability-based criterion. They also suggest that the L_1 loss function may be used for the goodness-of-fit term as being more robust. While they provide the updates to implement the L_1 loss function, they do not study its performance.

Marchetti and Zhou (2014) appear to be the first to focus on the MCP in solution path clustering. While they acknowledge the SCAD and truncated LASSO penalties, they prefer the MCP since it includes an explicit concavity parameter ω that is easy to separate from the penalization parameter λ. In fact, they vary the concavity parameter ω in their solution path construction in order to balance the bias-variance ratio in each of the clusters. They use a majorization-minimization algorithm (Lange, 2004) to compute the estimates. While they do not have a theoretical proof of convergence, they support their proposition with simulation experiments. They adopt the empirical approach from Fu and Zhou (2013) to select the number of clusters, which is similar to the elbow selection method.

Zhu, et. al. (2014) analyzed the conditions to recover the true clusters with convex

clustering. In particular, they prove that convex clustering can distinguish between two clusters under the condition that the distance between the two clusters is larger than some threshold which depends linearly on the size of the clusters and the ratio of the number of elements in the clusters.

Chi and Lange (2015) proposed ADMM and AMA (Alternating Minimization Algorithm; Tseng, 1991) algorithms for efficient computation of the global optimum for the convex clustering objective with in-depth complexity analysis. As opposed to the piecemeal approach of previous papers, they are the first to present a unified framework for computing the solution path for an arbitrary norm penalty. The ADMM and AMA algorithms appear to be the dominant choices for solving the convex clustering problem when looking at the more recent convex clustering papers in the literature.

Tan and Witten (2015) studied statistical properties of convex clustering. They provide an unbiased estimator of the degrees of freedom by viewing the problem as a penalized regression problem. They also derive bounds on the prediction error for convex clustering. By studying the dual problem of convex clustering, they show that it is closely related to single linkage hierarchical clustering and K-means clustering. This result is unsurprising, given that convex clustering can be developed as a convex relaxation of hierarchical clustering or K-means clustering, but their methods of proof are quite instructive. They also propose to use the extended BIC (Chen and Chen, 2008) to select the tuning parameter.

Ma and Huang (2017) employed the MCP and the SCAD penalty in their subgroup analysis and justify why a concave penalty is much more appropriate in this context.

They also appear to be the first to include a regression parameter within their framework. Specifically, their proposed objective function is

$$Q(m, \beta; \lambda) = \frac{1}{2} \sum_{i=1}^{n} (y_i - m_i - x_i^T \beta)^2 + \sum_{i<j} p(|m_i - m_j|, \lambda) \tag{2.17}$$

They develop an ADMM algorithm for the minimization of (2.17) and show that it converges to a local minimum. Furthermore, they show that the oracle estimator (the estimator obtained assuming the true cluster memberships are known) is a local solution of the objective function (2.17) with high probability. They also derive the order requirement of the minimum signal difference between groups such that the true clusters are recovered. They also proposed using the modified BIC (Wang, Li, and Leng, 2009) as a tuning parameter selection strategy with encouraging simulation results to support its use. In fact, it is Ma and Huang's (2017) work that is the foundation and springboard for our current project here.

Wang, et. al. (2016) proposed a version of convex clustering that is designed to be robust to outlier features. They "decompose the data matrix into a clustering structure component and a group sparse component that captures feature outliers" (Wang, et. al. 2016). They explicitly model the feature outliers in the data with a sparse matrix so that subtracting it from the original data matrix yields the "cleaned" data on which the clustering structure can be explored. This sparse matrix is estimated by adding another penalty term to the convex clustering objective function. Thus, two tuning parameters must be specified in addition to the weights. They develop a dedicated algorithm that is faster than the accelerated AMA algorithm, which Chi and Lange (2015) have shown is generally faster than the ADMM algorithm.

Similarly to Wang, et. al. (2016), Wang, et. al. (2018) proposed a sparse version of convex clustering where the goal is to simultaneously cluster observations and conduct feature selection. This is accomplished by adding an additional penalty term to the convex clustering objective function that controls the number of informative features. They develop ADMM and AMA algorithms in the same spirit of Chi and Lange (2015). Note that their method must specify weights as well as two tuning parameters, but they remark that the selection of important features is not sensitive to the particular clustering path, meaning that important features remain conspicuous in almost all clustering structures. They propose to select the tuning parameters with the stability measurement idea from Fang and Wang (2012) which is based on bootstrapping.

Shah and Koltun (2017) proposed an interesting variant of solution path clustering. Two distinguishing characteristics of their method is the use of a re-descending penalty function and using mutual kNN weights instead of the usual kNN. They claim that these two features allow heavily mixed and oddly shaped clusters to be discovered, and they have some simulation results to support this claim. They also demonstrate that their algorithm scales very efficiently to large datasets and high dimensions. They further extend their method to perform joint clustering and dimensionality reduction. It is an interesting article to read since it exhibits high-powered theory and results in a very short amount of space (6 pages).

Chapter 3

Solution Path Clustering with Robust Loss and Concave Penalty

3.1 Overview

Our main contribution is to extend the solution path clustering framework to include robust loss functions. The idea is that implementing a robust loss function will increase the overall clustering quality with improvements of the clustering location estimates, the clustering partition, and the estimated number of clusters. Extreme observations, outliers, and heavy-tailed error distributions will have less influence on the cluster results with a robust loss. These conditions can have a devastating effect when a least squares loss is used. Note that we consider only the univariate case in this chapter. Multivariate extensions can be found in Sections 4.1 and 4.2. Let $y_1, y_2, \ldots, y_n$ be a random sample to be represented by $m = (m_1, m_2, \ldots, m_n)$. The solution path clustering objective function that

was discussed in Chapter 2 can be expressed as

$$Q(m, \lambda) = \frac{1}{2} \sum_{i=1}^{n} (y_i - m_i)^2 + \sum_{i<j} p(|m_i - m_j|, \lambda), \tag{3.1}$$

where $p(\cdot, \lambda)$ is a penalty function to be specified. The penalty is a function of the tuning parameter λ, and possibly other parameters governing the behavior of the penalty. For example, the L_1 norm $p(|m_i - m_j|, \lambda) = \lambda |m_i - m_j|$ will recover the convex clustering criterion (2.1). One may also choose concave penalties, such as the truncated LASSO penalty, the SCAD penalty, or the MCP. While we have seen some variations on the choice of penalty function, all the solution path clustering schemes use the least squares loss for the goodness-of-fit term. It is well known that the least squares loss is very sensitive to outliers, where even a single extreme observation can ruin the estimation procedure. See Figure 3.1, for example. This motivates consideration of a *robust* loss function for the goodness-of-fit term. The new objective function we propose is

$$Q(m, \lambda) = \sum_{i=1}^{n} h(y_i - m_i) + \sum_{i<j} p_\omega\big(|m_i - m_j|, \lambda\big), \tag{3.2}$$

where $h(\cdot)$ is a (robust) loss function possibly parametrized by an additional threshold parameter r, and $p_\omega(\cdot, \lambda)$ is the MCP penalty function. We focus on the MCP penalty function in our work.

The goal is to augment the exploratory power of the solution path clustering framework to better recover the correct clustering structure. The robust loss and MCP penalty work together to mitigate the influence of outliers and minimize bias in the estimation of cluster centers, especially when the true distance between cluster centers is large. Since robust loss functions are not as affected by extreme observations as is the least squares loss,

it is more likely that larger observations will be clustered into a nearby cluster instead of remaining their own "cluster". For example, consider an outlier that has not been merged into a cluster yet. When a robust loss is used, the outlier is more easily merged into a cluster since whatever amount it adds to the loss term is not as large as the amount it adds to the penalty term. When a least squares loss is used, the λ value needs to be much larger in order to merge the same outlying observation since the amount it adds to the loss term can be very large compared to the amount it adds to the penalty term. This effect can be observed visually in solution path plots of Figure 3.1.

Introducing a new loss function means that we must develop the corresponding tools to implement it and study its properties. In Section 3.2.2, we introduce the IRLS-

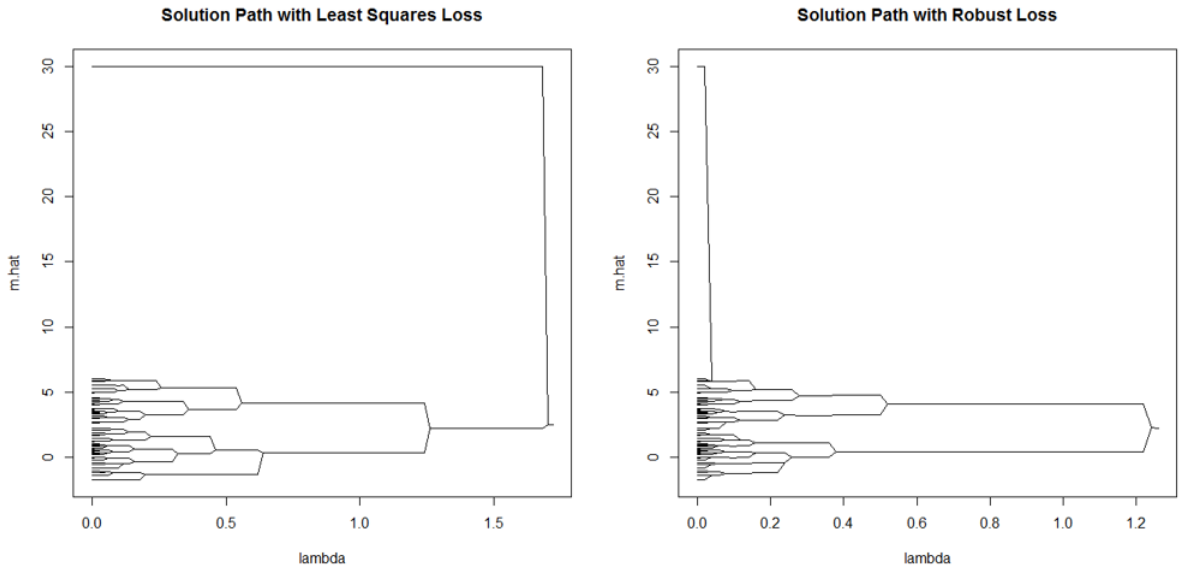


Figure 3.1: Two solution path plots using least squares loss and robust loss (Huber's approximation to absolute loss). 45 observations generated from $N(\mu = 0, \sigma^2 = 1)$, 45 are from $N(\mu = 4, \sigma^2 = 1)$ and adding in one outlier at 30.

ADMM algorithm to minimize our proposed objective function (3.2) and prove theoretically its convergence to a local minimum. We study consistency and oracle properties of the estimator in Section 3.4. Finally, in Section 3.5, we include simulation experiments to demonstrate the performance of our method and provide some preliminary results on choosing the number of clusters via modified BIC (Wang, Li, and Leng, 2009).

3.2 Computation

We develop the IRLS-ADMM algorithm (IRLS stands for Iteratively Reweighted Least Squares) here in a manner similar to Section 2.3. We show that if the loss function is solvable via an IRLS algorithm, then the IRLS-ADMM will converge to a local minimum. In fact, any loss function that admits an IRLS formulation or a majorizing surrogate can be used. Like before, the objective function (3.2) is difficult to minimize directly because the penalty function is not separable in the m_i's. Furthermore, minimizing robust loss functions generally involves solving a set of nonlinear equations, evoking a need for iterative methods (Holland and Welsch, 1977). We treat each of these issues in turn, using the variable splitting technique first, and then a IRLS formulation.

To circumvent the non-separability of the penalty function, we introduce a new set of parameters $\eta_{ij} = m_i - m_j$ and recast the optimization problem as

$$\begin{aligned} \min_{m \in \mathbb{R}^n} \quad & \sum_{i=1}^{n} h(y_i - m_i) + \sum_{i<j} p_\omega\big(|\eta_{ij}|, \lambda\big) \\ & \text{subject to } \Delta m - \eta = 0 \end{aligned} \tag{3.3}$$

Here, $\eta = (\eta_{ij}, i < j)$ is a vector of length $\binom{n}{2}$, and $\Delta = [(e_i - e_j), i < j]^T$ is the $\binom{n}{2} \times 2$ matrix where the e_i are $n \times 1$ vectors with the i-th element equal to 1 and the remaining

elements are 0. Let $\nu = (\nu_{ij}, i < j)$ be a $\binom{n}{2} \times 1$ vector of dual variables ν_{ij}. Encode the constraint by reformulating (3.3) with the augmented Lagrangian:

$$\begin{aligned} L_\beta(m, \eta, \nu) = \sum_{i=1}^{n} h(y_i - m_i) + \sum_{i<j} p_\omega(|\eta_{ij}|, \lambda) \\ + \sum_{i<j} \nu_{ij}(m_i - m_j - \eta_{ij}) + \frac{\beta}{2} \sum_{i<j} (m_i - m_j - \eta_{ij})^2 \end{aligned} \tag{3.4}$$

where β is a penalty parameter.

At this point, we would like to use an ADMM algorithm to minimize the objective function (3.4). Recall Step 1 of the ADMM updates where we must find the minimum of (3.4) with respect to m. In general, deriving the update with a robust loss function is not analytically tractable and iterative methods must be used. This leads us to the following section.

3.2.1 Deriving the IRLS Update

Here we derive an IRLS update which will aid us in the minimization of (3.4). In fact, we use the majorization-minimization (Lange, 2004) idea and minimize surrogate function. The particular surrogate function we choose recovers the familiar IRLS update which is analytically tractable. We try to use a "ground-up" approach to demonstrate the strategy behind this technique.

The main idea behind majorization-minimization is to find a surrogate function whose updates monotonically decrease the original loss function of interest. The surrogate function should bound (majorize) our original loss function. Thus, when we minimize the surrogate function, the update will yield a decrease in the original loss function. Usually,

we do not get to pick the original loss function, and it may be difficult to optimize directly. On the other hand, the surrogate function should be chosen so as to be easy to optimize (e. g. analytically tractable). This is the main benefit, since we are at liberty to choose *any* surrogate function as long as it satisfies basic properties, such as it majorizes the original loss function, and any stationary point of the surrogate function should simultaneously be a stationary point of the original loss function.

Assume that $L(m) = \sum_{i=1}^{n} h(y_i - m_i)$ is our objective function of interest. We wish to find a suitable surrogate function, say $u_0(\cdot)$. Since $L(m)$ is a sum, we can being by finding a surrogate for the function $h(\cdot)$. Assume that $h(\sqrt{e})$ is concave for $e \geq 0$. Note that most robust loss functions have this property (see Figure 3.2 below, and Aftab and Hartley, 2015). Now, for two points x and y, recall that any concave function $f(\cdot)$ satisfies

$$f(y) \leq f(x) + f'(x)(y - x). \tag{3.5}$$

We will apply (3.5) to $h(\sqrt{e})$ and use the points $e_i = (y_i - m_i)^2$ and $e_i^{(t)} = \left(y_i - m_i^{(t)}\right)^2$, where we have suppressed the notation that e_i and $e_i^{(t)}$ are functions of m_i and $m_i^{(t)}$, respectively. The superscript (t) represents the t-th iteration. Consider the observations y_i to be fixed. Then

$$\sum_{i=1}^{n} h(\sqrt{e_i}) \leq \sum_{i=1}^{n} h\left(\sqrt{e_i^{(t)}}\right) + \sum_{i=1}^{n} \frac{h'\left(\sqrt{e_i^{(t)}}\right)}{2\sqrt{e_i^{(t)}}} \left(e_i - e_i^{(t)}\right). \tag{3.6}$$

Note that the RHS of (3.6) majorizes the LHS. Now, plugging in $e_i = (y_i - m_i)^2$ and $e_i^{(t)} = \left(y_i - m_i^{(t)}\right)^2$ into (3.6) yields

$$L(m) = \sum_{i=1}^{n} h(y_i - m_i)$$

$$
\begin{aligned}
&= \sum_{i=1}^{n} h(\sqrt{e_i}) \\
&\leq \sum_{i=1}^{n} h\big(y_i - m_i^{(t)}\big) + \sum_{i=1}^{n} \frac{h'\big(y_i - m_i^{(t)}\big)}{2\big(y_i - m_i^{(t)}\big)} \Big[(y_i - m_i)^2 - \big(y_i - m_i^{(t)}\big)^2\Big] \qquad (3.7) \\
&:= u_0(m, m^{(t)}). \qquad \text{(assign definition)}
\end{aligned}
$$

In particular, see that

$$
\begin{aligned}
u_0\big(m, m\big) &= L(m), \\
u_0\big(m, m^{(t)}\big) &\geq L(m).
\end{aligned} \qquad (3.8)
$$

Also note that minimizing the RHS with respect to e is the same problem as

$$
\begin{aligned}
\min_{e} \sum_{i=1}^{n} \frac{h'\Big(\sqrt{e_i^{(t)}}\Big)}{2\sqrt{e_i^{(t)}}} e_i &= \min_{m} \sum_{i=1}^{n} \frac{h'(y_i - m_i^{(t)})}{2(y_i - m_i^{(t)})} (y_i - m_i)^2 \\
&= \min_{m} \sum_{i=1}^{n} w_i (y_i - m_i)^2
\end{aligned} \qquad (3.9)
$$

where we have made the assignment

$$
w_i := w(y_i, m_i^{(t)}) = \frac{h'\big(y_i - m_i^{(t)}\big)}{2(y_i - m_i^{(t)})} \qquad (3.10)
$$

Note that the last line of (3.9) is a weighted least squares problem which yields closed-form updates. Note also that the 2 in the denominator can be ignored since any weight matrix $W^* = cW$ any $c > 0$ will yield the same results as simply using W.

We have thus recovered the IRLS algorithm where the sequence of minimizers $\hat{m}$ depend upon weights that are functions of the most recent update of m.

Interestingly, notice that choosing the weights as in (3.10) satisfies

$$
\nabla_m h(|y_i - m_i|) = 0 \quad \text{iff} \quad \nabla_m w_i (y_i - m_i)^2 = 0 \qquad (3.11)
$$

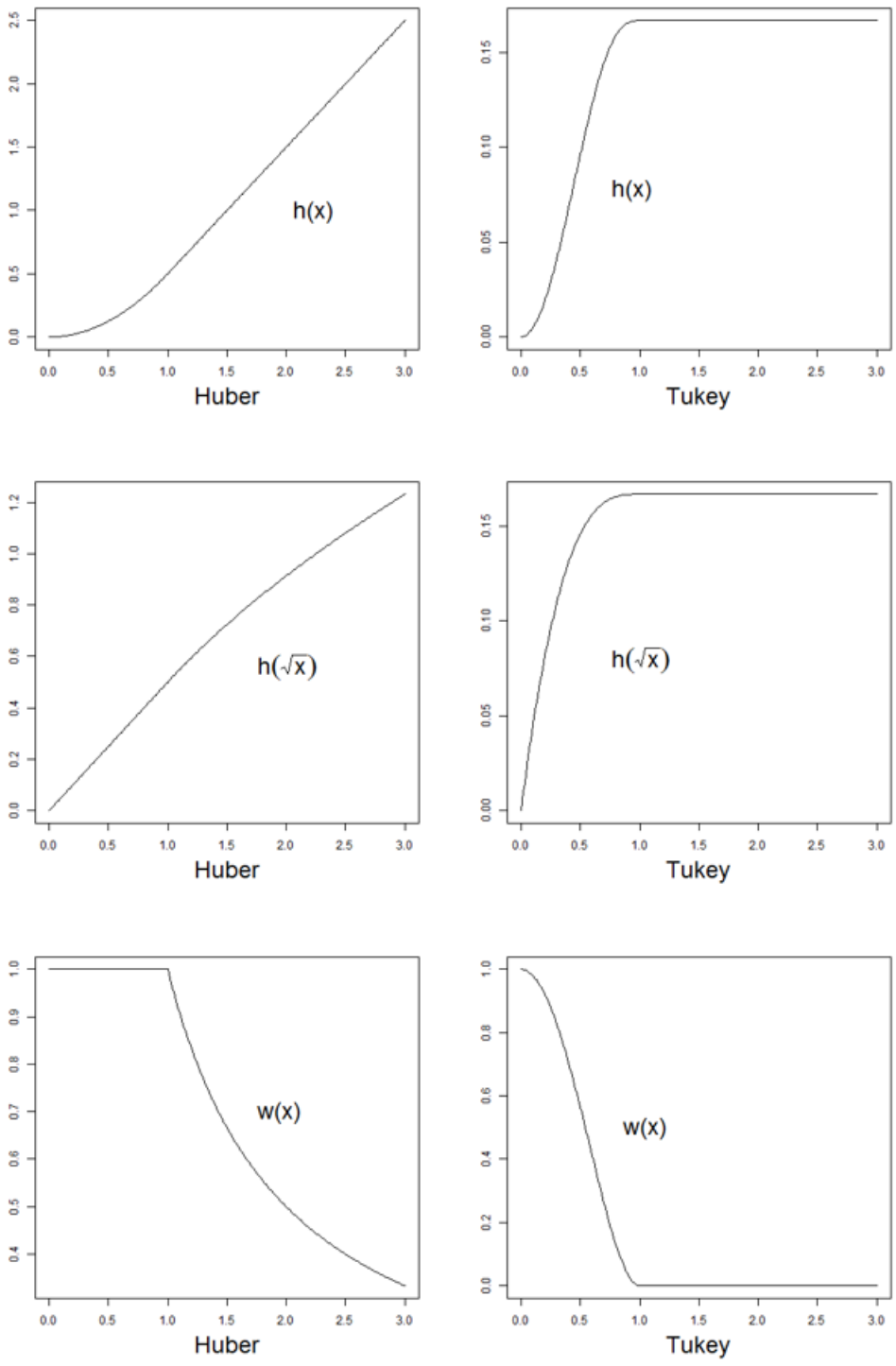


Figure 3.2: Plots of $h(x)$, $h(\sqrt{x})$, and $w(x)$ for the Huber and Tukey functions.

where ∇_m represents the gradient with respect to m. In fact, condition (3.11) is used in Aftab and Harley (2015) to derive the weight function (3.10).

3.2.2 IRLS-ADMM Algorithm

In light of the above discussion, we replace the robust loss function term in (3.4) with its WLS surrogate:

$$\begin{aligned} L^*_\beta(m, W, \eta, \nu) = \sum_{i=1}^{n} w_i (y_i - m_i)^2 + \sum_{i<j} p_\omega(|\eta_{ij}|, \lambda) \\ + \sum_{i<j} \nu_{ij}(m_i - m_j - \eta_{ij}) + \frac{\beta}{2} \sum_{i<j} (m_i - m_j - \eta_{ij})^2 \end{aligned} \tag{3.12}$$

Denote $W = \text{diag}\{w_1, w_2, \ldots, w_n\}$. It is convenient to rewrite (3.12) in matrix form:

$$\begin{aligned} L^*_\beta(m, W, \eta, \nu) = (y - m)^T W (y - m) + \sum_{i<j} p_\gamma(|\eta_{ij}|, \lambda) \\ + \nu^T(\Delta m - \eta) + \frac{\beta}{2} \big\| \Delta m - \eta \big\|^2 \end{aligned} \tag{3.13}$$

We propose to minimize (3.13) with an ADMM algorithm with an IRLS step inserted at the beginning of each iterate. At iteration t, the next cycle of updates consists of the following steps:

$$W^{(t+1)} := \text{diag}\big\{ w(y_i, m_i^{(t)}) \big\}_{i=1}^n \tag{Step 1}$$

$$m^{(t+1)} := \underset{m}{\text{argmin}}\, L^*_\beta(m, W^{(t+1)}, \eta^{(t)}, \nu^{(t)}) \tag{Step 2}$$

$$\eta^{(t+1)} := \underset{\eta}{\text{argmin}}\, L^*_\beta(m^{(t+1)}, W^{(t+1)}, \eta, \nu^{(t)}) \tag{Step 3}$$

$$\nu^{(t+1)} := \nu^{(t)} + \beta(\Delta m^{(t+1)} - \eta^{(t+1)}) \tag{Step 4}$$

Note that if the weight update step is omitted, it reduces to the standard ADMM algorithm steps.

In what follows, we use "hat" to denote the update and it is understood that any update is calculated using the current estimate of the other parameters.

The Step 2 update $m^{(t+1)}$ can be found in the usual way by setting the gradient equal to zero and solving. The update is

$$\hat{m} = (W + \beta\Delta^T\Delta)^{-1}[Wy + \beta\Delta^T(\eta - \beta^{-1}\nu)]$$

Note that $W + \beta\Delta^T\Delta = A - \beta\mathbf{1}\mathbf{1}^T$ where $A = W + n\beta I_n$ where $\mathbf{1}$ is a $n \times 1$ vector of ones. Then we can use the Sherman-Morrison formula for a fast calculation of the inverse:

$$(W + \beta\Delta^T\Delta)^{-1} = A^{-1} + \frac{\beta A^{-1}\mathbf{1}\mathbf{1}^T A^{-1}}{1 - \beta\mathbf{1}^T A^{-1}\mathbf{1}}$$

and A^{-1} is also easy to calculate since A is a diagonal matrix. Avoiding a numerical routine for calculating this matrix inverse saves appreciable computational time since we will need to calculate it within every iteration to reflect the updated weight matrix.

The Step 3 update is exactly the same as in Section 2.3:

$$\hat{\eta}_{ij} = \begin{cases} \mathrm{ST}\Big(\delta_{ij}, \frac{\lambda}{\beta}\Big) & \text{if} \quad |\delta_{ij}| \le \gamma\lambda \\ \delta_{ij} & \text{if} \quad |\delta_{ij}| > \gamma\lambda \end{cases} \tag{3.14}$$

where

$$\mathrm{ST}(\delta, c) = \Big(1 - \frac{c}{|\delta|}\Big)_+ \delta$$

is the soft thresholding operator.

The Step 4 update is also exactly the same as in Section 2.3.

We use the same initial values and warm start strategy as described in Section 2.3. The only addition is to use $W^{(0)} = \mathrm{diag}\big\{w(y_i, m_i^{(0)})\big\}$.

3.2.3 Convergence of IRLS-ADMM

For the t-th iteration, denote the primal residuals as $r^{(t)} = \Delta m^{(t)} - \eta^{(m)}$ and the dual residuals as $s^{(t)} = \beta \Delta^T (\eta^{(t+1)} - \eta^{(t)})$. We show that the ADMM iterates defined in Section 3.2.2 for

$$\begin{aligned} L^*_\beta(m, W, \eta, \nu) = (y-m)^T W (y-m) + \sum_{i<j} p_\gamma(|\eta_{ij}|, \lambda) \\ + \nu^T(\Delta m - \eta) + \frac{\beta}{2} \big\| \Delta m - \eta \big\|^2 \end{aligned} \tag{3.15}$$

achieve primal and dual feasibility, as stated in Proposition 1 below. Furthermore, the stationary points of L^*_β correspond to stationary points of

$$\begin{aligned} L_\beta(m, \eta, \nu) = \sum_{i=1}^{n} h(y_i - m_i) + \sum_{i<j} p_\omega(|\eta_{ij}|, \lambda) \\ + \nu^T(\Delta m - \eta) + \frac{\beta}{2} \big\| \Delta m - \eta \big\|^2. \end{aligned} \tag{3.16}$$

In fact, minimizing the surrogate function L^*_β (Equation (3.15)) accomplishes the goal of minimizing the original objective function L_β (Equation (3.16)). The proof is similar to Ma and Huang (2017) with a key difference being that we must use a result from Razaviyayn, Hong, and Luo (2013) instead of Tseng (2001). We first show, stated in the Lemma below, that the assumptions needed to apply their result are satisfied in our situation. It is cumbersome to copy all of the conditions and their requisite definitions, so we reference the equation numbers and result names used in Razaviyayn, Hong, and Luo (2013).

Lemma 1 *Define* $u(m, \hat{m}, \eta) = u_0(m, \hat{m}) + f_1(m, \eta)$, *where*

$$u_0(m, \hat{m}) = \sum_{i=1}^{n} h\big(y_i - \hat{m}_i\big) + \sum_{i=1}^{n} \frac{h'\big(y_i - \hat{m}_i\big)}{2\big(y_i - \hat{m}\big)} \Big[(y_i - m_i)^2 - \big(y_i - \hat{m}_i\big)^2\Big].$$

Then $u(m, \hat{m}, \eta)$ *satisfies Assumption 2 from Razaviyayn, Hong, and Luo (2013).*

Proof. We need to show that (B1), (B2), (B3), and (B4) in Razaviyayn, Hong, and Luo (2013). We begin by showing that their Proposition 2 holds, which implies that (B1), (B2), and (B3) hold. Now, write $f(m, \eta) = f_0(m) + f_1(m, \eta, \nu)$, where

$$\begin{aligned} f_0(m) &= \sum_{i=1}^{n} h(y_i - m_i), \\ f_1(m, \eta) &= \sum_{i<j} p_\omega(|\eta_{ij}|, \lambda) + \nu^T(\Delta m - \eta) + \frac{\beta}{2}\big\|\Delta m - \eta\big\|^2. \end{aligned}$$

First, note that $f_0(m)$ is differentiable. Second, since $f_1(m, \eta)$ is convex when $\omega > \beta^{-1}$, the directional derivative exists at all points (see, for example, page 83 in Mordukhovich and Nam, 2014). Now consider $u(m, \hat{m}, \eta)$ defined in the statement of this Lemma. (Recall that this function comes from the development in Section 3.2.1 above). Note that

$$u_0(m, m)) = f_0(m)$$

$$u_0(m, \hat{m}) \geq f_0(m)$$

for any m and $\hat{m}$. We have just shown Proposition 2 from Razaviyayn, Hong, and Luo (2013). Thus, (B1), (B2), and (B3) hold. If the $h'(y_i - m_i)\big[2(y_i - m_i)\big]^{-1}$ are continuous, so that $u_0(m, \hat{m})$ is continuous in $(m, \hat{m})$, then Assumption 2 in Razaviyayn, Hong, and Luo (2013) holds. ■

Proposition 2 *Let $h(\sqrt{x})$ be concave and differentiable for $x \geq 0$, and let the weight function $w(e)$ in (3.23) be continuous and defined for $e \geq 0$. As $t \to \infty$, we have $\left\|r^{(t)}\right\|^2 \to 0$ and $\left\|s^{(t)}\right\|^2 \to 0$. Also, (M^*, E^*) is a stationary point for L^*_β (Equation (3.15)) if and only if it is a stationary point for L_β (Equation (3.16)).*

Proof. Since $\eta^{(t+1)}$ is the minimizer of a convex function, we have

$$L^*_\beta(m^{(t+1)}, \eta^{(t+1)}, \nu^{(t)}) \leq L^*_\beta(m^{(t+1)}, \eta, \nu^{(t)}) \tag{3.17}$$

for any other η. Now, define

$$Q^*(m, \eta) = (y - m)^T W (y - m) + \sum_{i<j} p_\gamma(|\eta_{ij}), \lambda),$$

where $W = \text{diag}(w_1, w_2, \ldots, w_n)$ is a function of the most current update of m. Define also

$$f^{(t+1)} = \inf_{\Delta m^{(t+1)} - \eta = 0} Q^*(m^{(t+1)}, \eta).$$

Since the infimum is taken over the set where $\Delta m^{(t+1)} - \eta = 0$, we also have

$$f^{(t+1)} = \inf_{\Delta m^{(t+1)} - \eta = 0} L^*_\beta(m^{(t+1)}, \eta, \nu^{(t)}).$$

Then by the definition of $\eta^{(t+1)}$, we can write

$$L^*_\beta(m^{(t+1)}, \eta^{(t+1)}, \nu^{(t)}) \leq f^{(t+1)}. \tag{3.18}$$

We will now extend (3.18) to a similar conclusion for s iterations. Let s be a positive integer. By repeated substitution, we have

$$\nu^{(t+s-1)} = \nu^{(t)} + \beta \sum_{i=1}^{s-1} (\Delta m^{(t+i)} - \eta^{(t+i)}). \tag{3.19}$$

Then by plugging in (3.19), we have

$$
\begin{aligned}
&L^*_\beta(m^{(t+s)}, \eta^{(t+s)}, \nu^{(t+s-1)}) \\
&\quad = Q^*(m^{(t+s)}, \eta^{(t+s)}) + \nu^{(t+s-1)T}(\Delta m^{(t+s)} - \eta^{(t+s)}) + \frac{\beta}{2}\left\|\Delta m^{(t+s)} - \eta^{(t+s)}\right\|^2 \\
&\quad = Q^*(m^{(t+s)}, \eta^{(t+s)}) + \nu^{(t)T}(\Delta m^{(t+s)} - \eta^{(t+s)}) \\
&\qquad + \beta \sum_{i=1}^{s-1} (\Delta m^{(t+i)} - \eta^{(t+i)})^T (\Delta m^{(t+s)} - \eta^{(t+s)}) + \frac{\beta}{2}\left\|\Delta m^{(t+s)} - \eta^{(t+s)}\right\|^2 \\
&\quad \leq f^{(t+s)}.
\end{aligned}
\tag{3.20}
$$

Note that minimizing $u(m, \hat{m}, \eta)$ (defined in Lemma 1) with respect to m is equivalent to minimizing L^*_β (defined in (3.15)) with respect m. Their minimizations are also clearly equivalent with respect to η. Hence, the updates for $u(m, \hat{m}, \eta)$ are equivalent to the updates for L^*_β, and so the sequence of updates generated by minimizing $u(m, \hat{m}, \eta)$ is equivalent to the sequence of updates generated by minimizing L^*_β. Hence, we can apply Lemma 1. Using the results from Theorem 2 in Razaviyayn, Hong, and Luo (2013), the sequence $(m^{(t)}, \eta^{(t)})$ has a limit point, which we denote by (m^*, η^*). Then we have

$$
f^* = \lim_{t\to\infty} f^{(t+1)} = \lim_{t\to\infty} f^{(t+s)} = \inf_{\Delta m^* - \eta = 0} Q(m^*, \eta).
$$

Recalling the development in (3.20), for all integers $s \geq 0$ we have

$$
\begin{aligned}
&\lim_{t\to\infty} L^*_\beta(m^{(t+s)}, \eta^{(t+s)}, \nu^{(t+s-1)}) \\
&\qquad = Q^*(m^*, \eta^*) + \lim_{t\to\infty} \nu^{(t)T}(\Delta m^* - \eta^*) \\
&\qquad\quad + \beta \sum_{i=1}^{s-1} (\Delta m^* - \eta^*)^T (\Delta m^* - \eta^*) + \frac{\beta}{2}\left\|\Delta m^* - \eta^*\right\|^2 && \text{(plug in limits)} \\
&\qquad = Q^*(m^*, \eta^*) + \lim_{t\to\infty} \nu^{(t)T}(\Delta m^* - \eta^*) \\
&\qquad\quad + \beta \sum_{i=1}^{s-1} \left\|\Delta m^* - \eta^*\right\|^2 + \frac{\beta}{2}\left\|\Delta m^* - \eta^*\right\|^2 && \text{(equivalent notation)}
\end{aligned}
$$

$$
\begin{aligned}
&= Q^*(m^*, \eta^*) + \lim_{t\to\infty} \nu^{(t)T}(\Delta m^* - \eta^*) \\
&\qquad + \beta(s-1)\left\|\Delta m^* - \eta^*\right\|^2 + \frac{\beta}{2}\left\|\Delta m^* - \eta^*\right\|^2 \quad \text{(summing } s-1 \text{ identical terms)} \\
&= Q^*(m^*, \eta^*) + \lim_{t\to\infty} \nu^{(t)T}(\Delta m^* - \eta^*) + \beta(s - \frac{1}{2})\left\|\Delta m^* - \eta^*\right\|^2 \qquad \text{(simplify)} \\
&\leq f^*. \qquad \text{(if } x_n \to x,\ y_n \to y, \text{ and } x_n \leq y_n \text{ each } n, \text{ then } x \leq y)
\end{aligned}
$$

Thus, we must have $\lim_{t\to\infty} \left\|r^{(t)}\right\|^2 = \left\|\Delta m^* - \eta^*\right\|^2 = 0$.

Now, since the update $m^{(t+1)}$ minimizes $L^*_\beta(m, \eta^{(t)}, \nu^{(t)})$, we have

$$\frac{\partial L^*_\beta(m^{(t+1)}, \eta^{(t)}, \nu^{(t)})}{\partial m} = 0.$$

Recall that

$$
\begin{aligned}
\nu^{(t+1)} &= \nu^{(t)} + \beta(\Delta m^{(t+1)} - \eta^{(t+1)}) \\
\implies \nu^{(t)} &= \nu^{(t+1)} - \beta(\Delta m^{(t+1)} - \eta^{(t+1)}) \qquad (3.21)
\end{aligned}
$$

Then we can write

$$
\begin{aligned}
0 &= \frac{\partial L^*_\beta(m^{(t+1)}, \eta^{(t)}, \nu^{(t)})}{\partial m} \\
&= 2Wm^{(t+1)} - 2Wy + \Delta^T\nu^{(t)} + \beta\Delta^T(\Delta m^{(t+1)} - \eta^{(t)}) \qquad \text{(calculate derivative)} \\
&= 2Wm^{(t+1)} - 2Wy + \Delta^T\left[\nu^{(t)} + \beta(\Delta m^{(t+1)} - \eta^{(t)})\right] \qquad \text{(factor out } \Delta^T) \\
&= 2Wm^{(t+1)} - 2Wy \\
&\qquad + \Delta^T\left[\nu^{(t+1)} - \beta(\Delta m^{(t+1)} - \eta^{(t+1)}) + \beta(\Delta m^{(t+1)} - \eta^{(t)})\right] \qquad \text{(plug in (3.21))} \\
&= 2Wm^{(t+1)} - 2Wy + \Delta^T\left[\nu^{(t+1)} + \beta(\eta^{(t+1)} - \eta^{(t)})\right] \qquad \text{(cancel terms)} \\
&= 2Wm^{(t+1)} - 2Wy + \Delta^T\nu^{(t+1)} + \beta\Delta^T(\eta^{(t+1)} - \eta^{(t)}) \qquad (3.22)
\end{aligned}
$$

Since (3.22) equals zero, re-arranging the terms yields the definition of the dual residuals:

$$s^{(t+1)} = \beta\Delta^T(\eta^{(t+1)} - \eta^{(t)}) = -(2Wm^{(t+1)} - 2Wy + \Delta^T\nu^{(t+1)}).$$

Now, see that

$$\begin{aligned}
&\lim_{t\to\infty} \frac{\partial L^*_\beta(m^{(t+1)}, \eta^{(t)}, \nu^{(t)})}{\partial m} \\
&\quad = \lim_{t\to\infty} \left\{2Wm^{(t+1)} - 2Wy + \Delta^T\nu^{(t)} + \beta\Delta^T(\Delta m^{(t+1)} - \eta^{(t)})\right\} \quad \text{(calculate derivative)} \\
&\quad = 2Wm^* - 2Wy + \Delta^T\nu^* + \beta\Delta^T(\Delta m^* - \eta^*) \quad \text{(apply limit)} \\
&\quad = 2Wm^* - 2Wy + \Delta^T\nu^* \quad \text{(since } \left\|\Delta m^* - \eta^*\right\|^2 = 0\text{)} \\
&\quad = 0.
\end{aligned}$$

Therefore, $\lim_{t\to\infty} s^{(t+1)} = 0$.

Finally, consider L^*_β and L_β (defined in (3.15) and (3.16), respectively). We only need to examine their first terms since all of their other terms agree. Recall the weight function (defined in (3.10)) may be written as

$$w(e) = \frac{h'(e)}{2e}. \tag{3.23}$$

Then

$$\begin{aligned}
0 &= \frac{\partial}{\partial m}\sum_{i=1}^n w_i^*\left(y_i - m_i^*\right)^2 \\
&= -2\sum_{i=1}^n w_i^*(y_i - m_i^*) \\
&= -2\sum_{i=1}^n \frac{h'\left(y_i - m_i^*\right)}{2(y_i - m_i^*)}(y_i - m_i^*) \quad \text{(plug in weight formula (3.23))} \\
&= -\sum_{i=1}^n h'\left(y_i - m_i^*\right) \quad \text{(cancel terms)}
\end{aligned}$$

$$= \frac{\partial}{\partial m} \sum_{i=1}^{n} h\big(y_i - m_i^*\big).$$

Thus, stationary points of L_β coincide with stationary points of L_β^*.

■

3.3 Choice of Loss Function

For ease of reference, we copy our proposed objective function (3.2) here:

$$Q(m, \lambda) = \sum_{i=1}^{n} h(y_i - m_i) + \sum_{i<j} p_\omega\big(|m_i - m_j|, \lambda\big), \tag{3.24}$$

where $h(\cdot)$ is the loss function, and $p_\omega(\cdot, \lambda)$ is the MCP penalty function. Recall also that the weights formula (3.10) for use in the IRLS-ADMM algorithm can be expressed as

$$w_i := w(y_i, m_i) = \frac{h'(y_i - m_i)}{(y_i - m_i)}. \tag{3.25}$$

Our primary motivation for this work is to employ a robust loss function for the goodness-of-fit term in (3.24) instead of the usual least squares loss. This is to guard against undue influence from outliers in the data, which can ruin the search for subgroups. For example, a random sample from a heavy-tailed distribution can have several extreme observations. If one were to use the usual least squares loss when such pathologies in the data are present, it may lead to insensible cluster estimates and cluster partitions. This motivates developing a robust version of solution path clustering.

A first natural choice might be to choose the absolute loss function $h(y_i - m_i) = |y_i - m_i|$ which leads to a median estimator. According to (3.25), the weights formula in

this case is

$$w_i = \frac{\text{sign}(y_i - m_i)}{(y_i - m_i)} = \frac{1}{|y_i - m_i|} \tag{3.26}$$

if $m_i \neq y_i$, and is undefined if $m_i = y_i$. From an algorithmic point of view, the singularity whenever $m_i = y_i$ is a removeable one (Aftab, Harley, and Trumpf, 2015). Another approach, the one which we adopt here, is to replace (3.26) with

$$w_i = \frac{1}{\max\left\{r, |y_i - m_i|\right\}}, \tag{3.27}$$

where $r > 0$ is a regularization term chosen to be small, such as $r = 0.0001$. Interestingly, r can be interpreted as the threshold parameter in the Huber loss function. The Huber loss (Huber, 1964) can be written

$$h(y_i - m_i; r) = \begin{cases} \frac{1}{2}(y_i - m_i)^2 & \text{if} \quad |y_i - m_i| \leq r \\ r|y_i - m_i| - \frac{1}{2}r^2 & \text{if} \quad |y_i - m_i| > r, \end{cases} \tag{3.28}$$

which leads to a weights formula

$$w_i = \begin{cases} 1 & \text{if} \quad |y_i - m_i| \leq r \\ \dfrac{r}{|y_i - m_i|} & \text{if} \quad |y_i - m_i| > r. \end{cases} \tag{3.29}$$

It can be seen that weight functions (3.26) and (3.29) are equivalent since the weights need to be known only up to a common constant. Thus, the Huber loss with threshold parameter r chosen to be small is a smoother, albeit approximate, version of the absolute loss. For more details and an application of the approximation, the interested reader is referred to Fountoulakis and Gondzio (2016).

The typical context in which the Huber loss function is used is when one wishes the balance the benefits of absolute loss and least squares loss. A quick examination of (3.28) shows this to be the case, where the threshold parameter r controls the amount of balancing between the two types of loss functions. Usually, the Huber loss is used when one wishes to retain the efficiency of the least squares loss (if the errors happen to be Gaussian) while mixing in the robustness of absolute loss. This motivates a choice of the tuning parameter r. We can choose r so that the resulting estimator still has a relatively high efficiency if the data were really Gaussian. To give 95% efficiency in the Gaussian case, we use $r = 1.345\sigma$, where σ scales the data (Fox and Weisberg, 2013). A commonly used robust estimate of scale is

$$\hat{\sigma} = \frac{\text{MAR}}{0.6745}$$

where MAR is the median absolute residual.

Another common robust loss function is the Tukey biweight function (Tukey, 1960), which can be written as

$$h(y_i - m_i, r) = \begin{cases} \dfrac{r^2}{6}\left\{1 - \left[1 - \left(\dfrac{y_i - m_i}{r}\right)^2\right]^3\right\} & \text{if} \quad |y_i - m_i| \leq r \\ \dfrac{r^2}{6} & \text{if} \quad |y_i - m_i| > r, \end{cases} \tag{3.30}$$

which leads to weights formula

$$w_i = \begin{cases} \left[1 - \left(\dfrac{y_i - m_i}{r}\right)^2\right] & \text{if} \quad |y_i - m_i| \leq r \\ 0 & \text{if} \quad |y_i - m_i| > r. \end{cases} \tag{3.31}$$

Note that the Tukey biweight function is not convex. Whether the loss function of interest is convex or not turns out to be an important distinction when developing the asymptotic

theory. The Tukey loss function is in the class of re-descending M-estimators (Huber, 1981). In particular, it has a "drop-off" point where once an observation is beyond a certain threshold, its influence on the estimator does not change the further beyond to the extremity that it goes. Hence, the threshold parameter r controls the amount of resistance to outliers. Recall that robustness comes at the cost of efficiency. To give 95% efficiency in the Gaussian case, we use $r = 4.685\sigma$ where σ scales the data (Fox and Weisberg, 2013).

3.4 Theoretical Properties

For any vector $x = (x_1, x_2, \ldots, x_n)$, let $\|x\|_\infty$ denote the maximum norm of x, defined by $\|x\|_\infty = \max\big(|x_1|, |x_2|, \ldots, |x_n|\big)$. For any positive sequences of real numbers a_n and b_n, let $a_n \gg b_n$ denote $a_n^{-1} b_n \to 0$ and $a_n \ll b_n$ denote $a_n b_n^{-1} \to 0$. Let $\lambda_{\min}(X)$ and $\lambda_{\max}(X)$ denote the minimum and maximum eigenvalues of a matrix X, respectively. Let n_k denote the number of elements in G_k and let $n_{\min} = \min_{1 \le k \le K}\{n_k\}$.

Consider the model

$$y_i = z_i^T \alpha + e_i, \qquad i = 1, 2, \ldots, n \tag{3.32}$$

where the e_i are independent and identically distributed error variables according to the density f. When studying oracle properties, we assume that the true memberships z_i are observed. The oracle estimator $\hat{\alpha}_n^{\text{or}}$ is defined as the solution to the estimating equation

$$\sum_{i=1}^{n} \phi(y_i - z_i^T \hat{\alpha}_n^{\text{or}})\, z_i = 0, \tag{3.33}$$

where $\phi = h'$. When the loss function $h(\cdot)$ is convex, then $\hat{\alpha}^{\text{or}}$ is unique. If $h(\cdot)$ is not convex, then define the oracle estimator $\hat{\alpha}^{or}$ as the global minimizer to (3.33). Denote

$\hat{m}^{or} = \{\hat{m}_1^{or}, \hat{m}_2^{or}, \ldots, \hat{m}_n^{or}\}$ where $\hat{m}_i^{or} = \sum_{j=1}^{K} \alpha_j^{or} I(i \in G_j)$ where $I(\cdot)$ is the indicator function. Let $D_n = n^{-1} \sum_{i=1}^{n} z_i z_i^T$. In our special case, since the z_i are categorical variables (there is a 1 in the j-th coordiante and 0's everywhere else), $D_n = \text{diag}(\hat{\pi}_1, \hat{\pi}_2, \ldots, \hat{\pi}_K)$ where each $\hat{\pi}_j = n^{-1} n_j$ where n_j is the number of elements in the j-th group.

The oracle estimator $\hat{\alpha}^{or}$ in (3.33) is an M-estimator. Stefanski and Boos (2002) provide an illuminating introduction to M-estimation. Huber's (1981) book is a classic. Serfling (Chapter 7, 1980) is also a good starting point to look at important results and references for M-estimation.

The following conditions are considered:

(D1) $0 < \liminf_n \lambda_{\min}(D_n) \leq \limsup_n \lambda_{\max}(D_n) < \infty$.

Thus, every component must have at least one observation.

(D2) ϕ, f, and f' are bounded with $0 < \gamma < \infty$ where

$$\gamma = -\int_{-\infty}^{\infty} \phi(r) f'(r) dr.$$

Note that γ is essentially $E\phi'(e)$ after integration by parts. However, using γ as it is expressed here does not require that ϕ have a derivative.

(D3) $\max_{i \leq n} \|z_i\|^2 = O(K)$ and $\sup_{\beta,\gamma \in S_m} \sum_{i=1}^{n} |z_i^T \beta|^2 |z_i^T|^2 = O(n)$, where $S_m = \{a \in R^K : \|a\| = 1\}$. In our special case, $\max_{i \leq n} \|z_i\|^2 = 1$, and $\sup_{\beta,\gamma \in S_m} \sum_{i=1}^{n} |z_i^T \beta|^2 |z_i^T|^2 = n$, so (D3) is satisfied quite trivially in our case.

(D4) The loss function h is convex, and $|\phi(x)| \leq |x|$.

The first part here is a sufficient condition for the existence of a unique minimizer in the oracle case. The second part gives

$$|h'(x)| = |\phi(x)| \leq |x| = \Big|\frac{d}{dx}\Big(\frac{1}{2}x^2\Big)\Big|,$$

so that the derivative of the chosen loss function is smaller in magnitude than the derivative of the least squares loss function. This makes intuitive sense as we are aiming towards robust loss functions. (D4) is satisfied by the Huber function, for example.

(D5) $E\big[\phi(e_i)\big] = 0$ and $E\big[\phi^2(e_i)\big] < \infty$.

The first statement here defines the "true parameter." The second statement ensures a non-degenerate limiting normal distribution.

(D6) The error terms e_i are independent and identically distributed according to density f.

(D7) $E\big(|e_i|^\zeta\big) < \infty$ some $\zeta > 1$.

The value of ζ in some sense determines the heaviness of the tails of the error dsitribution. For example, the normal distribution and the t distribution with degrees of freedom greater than 1 satisfy this condition. The Cauchy distribution does not.

(D8) The penalty function $p_\omega(t, \lambda)$ is symmetric in t, and it is non-decreasing and concave for $t \in [0, \infty)$. There exists a finite constant $\kappa > 0$ such that $\rho(t)$ is constant for all $t \geq \kappa\lambda$, and $\rho(0) = 0$. Finally, $\rho'(t)$ exists and is continuous except for a finite number of t and $\rho'(0+) = 1$.

Theorem 3 *(Consistency and Asymptotic Normality) Assume that the conditions (D1) – (D6) are fulfilled. If* $K^3(\log K)^2 = o(n)$*, then*

$$\|\hat{\alpha}_n^{or} - \alpha_0\|^2 = O_p(K/n). \tag{3.34}$$

Furthermore, for any $a \in \mathbb{R}^K$*,*

$$\frac{\sqrt{n}\, a^T\big(\hat{\alpha}_n^{or} - \alpha^0\big)}{\sigma_n(a)} \to N(0,1)$$

as $n \to \infty$*, where* $\sigma_n^2(a) = \gamma^{-2}\left[E\phi^2(e)\right] a^T D_n^{-1} a$*.*

Proof. This is Corollary 2.1 of He and Shao (2000). ■

Note that for this consistency result, we can state: for any $\epsilon > 0$, there exists a constant M_0 such that for n large enough,

$$P\big(\|\hat{\alpha}_n^{or} - \alpha_0\|^2 \leq M_0 K/n\big) > 1 - \epsilon. \tag{3.35}$$

If we directly use the Bahadur representation (Theorem 2.2 of the same He and Shao (2000) paper), we can obtain a finite sample property instead of just an asymptotic statement (given an ϵ, if we go far out enough in the sequence, etc...).

Theorem 4 *(Finite sample property) Assume that the conditions (D1) – (D6) are fulfilled. Let* $1/2 < a < 1$ *and define* $\|r_n\| = o_p(n^{-1/2})$*. If* $K^3(\log K)^2 = o(n)$ *and* $n^{-1} n_{min} \to \pi_{min}$ *where* $0 < \pi_{min} \leq 1$*, then we have*

$$P\Big(\big\|\hat{\alpha}_n^{or} - \alpha^0\big\|_\infty \leq g_n\Big) > 1 - \frac{E\phi^2(e)}{n^{2a-1}}, \tag{3.36}$$

where

$$g_n = \frac{n^a}{|\gamma| n_{\min}} + \|r_n\|_\infty. \tag{3.37}$$

Proof. Define $Q_n = nD_n = \sum_{i=1}^n z_i z_i^T$. We apply Theorem 2.2 of He and Shao (2000) and write

$$\hat{\alpha}_n^{\text{or}} - \alpha^0 = -(\gamma Q_n)^{-1} \sum_{i=1}^n \phi(e_i) z_i + r_n. \tag{3.38}$$

An application of the Triangle Inequality gives

$$\left\|\hat{\alpha}_n^{\text{or}} - \alpha^0\right\|_\infty \leq \left\|(\gamma Q_n)^{-1} \sum_{i=1}^n \phi(e_i) z_i\right\|_\infty + \|r_n\|_\infty. \tag{3.39}$$

Then

$$\begin{aligned}
P\Big(\left\|\hat{\alpha}_n^{\text{or}} - \alpha_0\right\|_\infty > g_n\Big) &= P\Big(\left\|\hat{\alpha}_n^{\text{or}} - \alpha_0\right\|_\infty > \frac{n^a}{|\gamma| n_{\min}} + \|r_n\|_\infty\Big) && \text{(plug in } g_n\text{)} \\
&\leq P\Big(\left\|(\gamma Q_n)^{-1} \sum_{i=1}^n \phi(e_i) z_i\right\|_\infty + \|r_n\|_\infty > \frac{n^a}{|\gamma| n_{\min}} + \|r_n\|_\infty\Big) \\
& && \text{(by (3.39))} \\
&= P\Big(\left\|(\gamma Q_n)^{-1} \sum_{i=1}^n \phi(e_i) z_i\right\|_\infty > \frac{n^a}{|\gamma| n_{\min}}\Big). && \text{(the } r_n\text{'s cancel)}
\end{aligned}$$

Since $n_{\min}$ is the smallest diagonal element of the diagonal matrix Q_n, for any $x \in \mathbb{R}^K$ we have

$$\left\|n_{\min}^{-1} x\right\|_\infty \geq \left\|Q_n^{-1} x\right\|_\infty. \tag{3.40}$$

Thus,

$$
\begin{aligned}
&P\Big(\big\|(\gamma Q_n)^{-1}\sum_{i=1}^{n}\phi(e_i)z_i\big\|_\infty > \frac{n^a}{|\gamma|n_{\min}}\Big) && \text{(re-writing from above)}\\
&\quad\le P\Big(\big\|(\gamma n_{\min})^{-1}\sum_{i=1}^{n}\phi(e_i)z_i\big\|_\infty > \frac{n^a}{|\gamma|n_{\min}}\Big) && \text{(by (3.40))}\\
&\quad= P\Big(\big|(\gamma n_{\min})^{-1}\big| \times \big\|\sum_{i=1}^{n}\phi(e_i)z_i\big\|_\infty > \frac{n^a}{|\gamma|n_{\min}}\Big) && \text{(property of norms)}\\
&\quad= P\Big(\big\|\sum_{i=1}^{n}\phi(e_i)z_i\big\|_\infty > n^a\Big) && \text{(cancel terms)}\\
&\quad= P\left(\max_{1\le j\le K}\Big(\big|\sum_{i\in G_j}\phi(e_i)\big|\Big) > n^a\right) && \text{(by definition of } \|\cdot\|_\infty)\\
&\quad\le \sum_{j=1}^{K} P\Big(\big|\sum_{i\in G_j}\phi(e_i)\big| > n^a\Big). && \text{(by union bound)}
\end{aligned}
$$

By (D5) and (D6), the random variables $\phi(e_i)$ are independent and identically distributed with $E\big[\phi(e)\big] = 0$ and $E\big[\phi^2(e)\big] < \infty$. Then for each $j = 1, 2, ..., K$, we have

$$
\begin{aligned}
P\left(\Big|\sum_{i\in G_j}\phi(e_i)\Big| > n^a\right) &\le \frac{E\Big(\big|\sum_{i\in G_j}\phi(e_i)\big|^2\Big)}{n^{2a}} && \text{(by Markov's Inequality)}\\
&= \frac{n_j\big[E\phi^2(e)\big]}{n^{2a}}, \qquad \text{(i.i.d., } E\big[\phi(e)\big] = 0\text{, so cross-terms are zero)}
\end{aligned}
$$

where n_j denotes the number of elements in G_j. Summing across the j index, we have

$$
\begin{aligned}
\sum_{j=1}^{K} P\Big(\big|\sum_{i\in G_j}\phi(e_i)\big| > n^a\Big) &\le \frac{n\big[E\phi^2(e)\big]}{n^{2a}} && \text{(since } \textstyle\sum_{j=1}^{K} n_j = n)\\
&= \big[E\phi^2(e)\big]n^{1-2a}.
\end{aligned}
$$

Thus, $P\Big(\big\|\hat{\alpha}_n^{\text{or}} - \alpha^0\big\|_\infty > g_n\Big) \le \big[E\phi^2(e)\big]n^{1-2a}$, and since $1 < 2a$ by theorem assumption, we have $g_n \to 0$ and $\big[E\phi^2(e)\big]n^{1-2a} \to 0$ as $n \to \infty$. This confirms our consistency result.

We can also verify the asymptotic normality. From (3.38), we have

$$\begin{aligned}
\hat{\alpha}_n^{\text{or}} - \alpha^0 &= -(\gamma Q_n)^{-1} \sum_{i=1}^{n} \phi(e_i) z_i + r_n \\
&= -(\gamma n^{-1} Q_n)^{-1} \, \frac{1}{n} \sum_{i=1}^{n} \phi(e_i) z_i + r_n && \text{(multiply by } n/n\text{)} \\
&= -(\gamma D_n)^{-1} \, \frac{1}{n} \sum_{i=1}^{n} \phi(e_i) z_i + r_n. && \text{(since } D_n = n^{-1} Q_n\text{)}
\end{aligned}$$

Let $a \in \mathbb{R}^K$. Multiplying both sides of the above by $\sqrt{n}\, a^T$ yields

$$\sqrt{n}\, a^T \big(\hat{\alpha}_n^{\text{or}} - \alpha^0\big) = -a^T \big(\gamma D_n\big)^{-1} \frac{1}{\sqrt{n}} \sum_{i=1}^{n} \phi(e_i) z_i + \sqrt{n}\, a^T r_n. \tag{3.41}$$

From (D5) and (D6), the e_i are independent and identically distributed with $E[\phi(e)] = 0$ and $\text{Var}[\phi(e)] = E[\phi^2(e)] < \infty$. Then the expectation of (3.41) tends to zero in the limit since $\sqrt{n}\, a^T r_n \to 0$. The variance of (3.41), denoted $\sigma_n^2(a)$, is

$$\begin{aligned}
\sigma_n^2(a) &= a^T (\gamma D_n)^{-1} \left(\frac{1}{n} \sum_{i=1}^{n} \text{Var}\big[\phi(e_i)\big] z_i z_i^T \right) (\gamma D_n)^{-1} a && \text{(calculate variance)} \\
&= a^T (\gamma D_n)^{-1} \left(\frac{E\phi^2(e)}{n} \sum_{i=1}^{n} z_i z_i^T \right) (\gamma D_n)^{-1} a && \text{(using } \text{Var}[\phi(e_i)] = E\big[\phi^2(e_i)\big]\text{)} \\
&= a^T (\gamma D_n)^{-1} \big[E\phi^2(e)\, D_n \big] (\gamma D_n)^{-1} a && \text{(using } n^{-1} \textstyle\sum_{i=1}^{n} z_i z_i^T = D_n\text{)} \\
&= \gamma^{-2} \big[E\phi^2(e) \big] a^T (D_n)^{-1} a. && \text{(simplify)}
\end{aligned}$$

Then by the Central Limit Theorem, as $n \to \infty$, we have

$$\frac{\sqrt{n}\, a^T \big(\hat{\alpha}_n^{\text{or}} - \alpha^0\big)}{\sigma_n(a)} \to N\big(0, 1\big). \tag{3.42}$$

■

Notice that the theorem requires $n^{-1} n_{\min} \to \pi_{\min} \in (0, 1]$. This means that $n_{\min}$ is of the same order as n which ensures that $g_n \to 0$. Thus, the z_i can be seen as being

drawn from a multinomial distribution with probability parameter $\pi = (\pi_1, \pi_2, \ldots, \pi_K)$, where $\sum_{j=1}^{K} \pi_j = 1$ and $\min_{1 \le j \le K}\{ \pi_j \} = \pi_{\min} > 0$. This is a somewhat strict requirement on the order of growth for the smallest component. For example, the assumption in Ma and Huang (2017) on the smallest component's growth is weaker.

Theorem 5 *Suppose conditions (D1) – (D8) hold. Let $\zeta^{-1} < d < a$ where ζ comes from (D7) and $1/2 < a < 1$. Denote $b_n = \min_{k \neq k'} |\alpha_k^0 - \alpha_{k'}^0|$ to be the true minimal difference between two subgroups. If $\kappa\lambda < b_n$ and $g_n \ll \lambda$ where κ is given in (D8) and g_n is given in (3.37), then there exists a local minimizer $\hat{m}(\lambda)$ of the objective function $Q_n(\cdot)$ (defined in (3.2)) satisfying*

$$P[\hat{m}(\lambda) = \hat{m}^{or}] \to 1. \tag{3.43}$$

Proof. Define $\rho(t) = \lambda^{-1} p_\omega(t, \lambda)$, and let

$$\begin{aligned} L_n(m) &= \sum_{i=1}^{n} h(y_i - m_i), \quad P_n(m) = \lambda \sum_{i<j} \rho(|m_i - m_j|), \\ L_n^G(\alpha) &= \sum_{i=1}^{n} h(y_i - z_i^T \alpha), \; P_n^G(\alpha) = \lambda \sum_{k<k'} n_k n_{k'} \rho(|\alpha_k - \alpha_{k'}|), \end{aligned} \tag{3.44}$$

and

$$\begin{aligned} Q_n(m) &= L_n(m) + P_n(m), \\ Q_n^G(\alpha) &= L_n^G(\alpha) + P_n^G(\alpha). \end{aligned} \tag{3.45}$$

Define M_G to be a subspace of $\mathbb{R}^n$ by

$$M_G = \{ m \in \mathbb{R}^n : m_i = m_j \text{ for any } i, j \in G_k, 1 \le k \le K \}.$$

Let $T\colon M_G \to \mathbb{R}^K$ be the mapping such that $T(m) = \alpha$, where the kth coordinate is equal to the common value of m_i for $i \in G_k$. Let $T^*\colon \mathbb{R}^n \to \mathbb{R}^K$ be the mapping defined by

$$T^*(m) = \Big\{ n_k^{-1} \sum_{i \in G_k} m_i \Big\}_{k=1}^{K}.$$

Thus, it is easy to see that $T(m) = T^*(m)$ whenever $m \in M_G$.

For every $m \in M_G$, we have $L_n(m) = L_n^G\big(T(m)\big)$ and $P_n(m) = P_n^G\big(T(m)\big)$. For every $\alpha \in \mathbb{R}^K$, we have $L_n\big(T^{-1}(\alpha)\big) = L_n^G(\alpha)$ and $P_n\big(T^{-1}(\alpha)\big) = P_n^G(\alpha)$. Thus,

$$Q_n(m) = Q_n^G\big(T(m)\big) \quad \text{and} \quad Q_n^G(\alpha) = Q_n\big(T^{-1}(\alpha)\big) \tag{3.46}$$

hold for $m \in M_G$ and $\alpha \in \mathbb{R}^K$.

Recall that α^0 is regarded as the "true parameter," and write $m^0 = T^{-1}(\alpha^0)$. Consider the neighborhood of m^0 defined by

$$\Theta = \big\{\, m \in R^n : \big\|m - m^0\big\|_\infty \le g_n \,\big\} \tag{3.47}$$

where g_n is given by (3.37). By Theorem 4, there is an event, say E_1, on which $\big\|\hat{m}^{or} - m^0\big\|_\infty \le g_n$ and $P(E_1) > 1 - \big[E\phi^2(e)\big]n^{1-2a}$. Thus, $\hat{m}^{or} \in \Theta$ on E_1.

For any $m \in \mathbb{R}^n$, denote $m^* = T^{-1}\big(T^*(m)\big)$, so the number of unique values in m^* is the number of subgroups. We will show that $\hat{m}^{or}$ is a strictly local minimizer of the objective function $Q_n(m)$ (defined in (3.45)) with probability approaching one, through the following two steps and their sub-steps:

1. On event E_1, for any $m \in \Theta$ such that its $m^* \neq \hat{m}^{or}$, we have $Q_n(\hat{m}^{or}) < Q_n(m^*)$.

(a) Show that for any $m \in \Theta$, the penalty term in the objective function $Q_n(m^*)$ is a constant not depending on m.

(b) Since $\hat{m}^{or} \in \Theta$ on event E_1, then the penalty terms for $Q_n(\hat{m}^{or})$ and for $Q_n(m^*)$ are both a constant and identical.

(c) Since their penalty terms are equal, it remains to compare the goodness-of-fit terms. This is easy since $\hat{m}^{or}$ is the global minimizer for $Q_n(\cdot)$.

2. There is an event, say E_2, with $P(E_2) > \left(1 - \frac{c_1}{n^{dc_2}}\right)^n$. On event $E_1 \cap E_2$, there is a neighborhood, say Θ_n, of $\hat{m}^{or}$ such that for any $m \in \Theta \cap \Theta_n$ with n large enough, we have $Q_n(m^*) \leq Q_n(m)$.

   (a) To compare $Q_n(m)$ and $Q_n(m^*)$, calculate the Taylor expansion of $Q_n(m)$ about m^* to obtain $Q_n(m) = Q_n(m^*) + \Gamma_1 + \Gamma_2$.

   (b) Derive a lower bound on Γ_2.

   (c) Derive a lower bound on Γ_1.

   (d) Use the lower bounds to show $\Gamma_1 + \Gamma_2 \geq 0$. Thus, $Q_n(m) - Q_n(m^*) = \Gamma_1 + \Gamma_2 \geq 0$.

Then Steps 1 and 2 together say that $Q_n(\hat{m}^{or}) < Q_n(m^*) \leq Q_n(m)$ for any $m \in \Theta \cap \Theta_n$ such that $m \neq \hat{m}^{or}$, meaning that $\hat{m}^{or}$ is a strict local minimizer of $Q_n(m)$ on $E_1 \cap E_2$. Furthermore,

$$
\begin{aligned}
P(E_1 \cap E_2) &= 1 - P\Big(E_1^C \cup E_2^C\Big) && \text{(complement and de Morgan's law)} \\
&\geq 1 - \Big[P\big(E_1^C\big) + P\big(E_2^C\big)\Big] && \text{(union bound)} \\
&\geq 1 - \left\{\big[E\phi^2(e)\big] n^{1-2a} + \left[1 - \Big(1 - \frac{c_1}{n^{dc_2}}\Big)^n\right]\right\} && \text{(plug in probabilities)}
\end{aligned}
$$

for sufficiently large n. Since $1 < 2a$ and $1 < c_2 d$ by Theorem assumption, then $\big[E\phi^2(e)\big]n^{1-2a} \to 0$ and $\left(1 - \frac{c_1}{n^{dc_2}}\right)^n \to 1$ as $n \to \infty$. Thus, $P(E_1 \cap E_2) \to 1$ as $n \to \infty$.

Proof of Step 1: Let $m \in \Theta$. We first show that $P_n^G\big(T^*(m)\big) = C_n$ some constant independent of m. Call $\alpha = T^*(m)$.

It suffices to show that $|\alpha_k - \alpha_{k'}| > \kappa\lambda$ for all $k \neq k'$, because then by (C5), $\rho(|\alpha_k - \alpha_k'|)$ is a constant which shows that $P_n^G\big(T^*(m)\big)$ is also a constant. Now, for any k and k' unequal, we have

$$\begin{aligned}
|\alpha_k^0 - \alpha_{k'}^0| &= |\alpha_k^0 - \alpha_k + \alpha_k - \alpha_{k'} + \alpha_{k'} - \alpha_{k'}^0| && \text{(add zero twice)} \\
&\leq |\alpha_k - \alpha_{k'}| + |\alpha_k^0 - \alpha_k| + |\alpha_{k'} - \alpha_{k'}^0| && \text{(triangle inequality)} \\
&\leq |\alpha_k - \alpha_{k'}| + 2\big\|\alpha - \alpha^0\big\|_\infty && \text{(twice the max} \geq \text{sum of any other two)}
\end{aligned}$$

Re-arranging, we can write this as

$$|\alpha_k - \alpha_{k'}| \geq |\alpha_k^0 - \alpha_{k'}^0| - 2\big\|\alpha - \alpha^0\big\|_\infty \tag{3.48}$$

Now, note that

$$\begin{aligned}
\big\|\alpha - \alpha^0\big\|_\infty &= \sup_k \left| \Big(\sum_{i \in G_k} \frac{m_i}{n_k} \Big) - \alpha_k^0 \right| && \text{(from } \alpha = T(^*m)\text{; by definition of } \|\cdot\|_\infty) \\
&= \sup_k \left| \sum_{i \in G_k} \frac{m_i - m_i^0}{n_k} \right| && \text{(since } m_i^0 = \alpha_k^0 \text{ for } i \in G_k) \\
&\leq \sup_k \sup_{i \in G_k} \big| m_i - m_i^0 \big| && \text{(taking the supremum over } i \text{ as well)} \\
&= \big\| m - m^0 \big\|_\infty && \text{(by definiton of } \|\cdot\|_\infty) \\
\implies -2\big\|\alpha - \alpha^0\big\|_\infty &\geq -2\big\| m - m^0 \big\|_\infty && (3.49)
\end{aligned}$$

Then for all k and k' unequal, we have

$$\begin{aligned}
|\alpha_k - \alpha_{k'}| &\geq |\alpha_k^0 - \alpha_{k'}^0| - 2\big\|\alpha - \alpha^0\big\|_\infty && \text{(going from (3.48))} \\
&\geq |\alpha_k^0 - \alpha_{k'}^0| - 2\big\|m - m^0\big\|_\infty && \text{(from (3.49))} \\
&\geq |\alpha_k^0 - \alpha_{k'}^0| - 2g_n && \text{(since } m \in \Theta\text{)} \\
&\geq b_n - 2g_n && \text{(by definition of } b_n\text{)} \\
&> \kappa\lambda && (3.50)
\end{aligned}$$

since $b_n > \kappa\lambda$ by theorem assumption. Thus, we have

$$\begin{aligned}
Q_n^G\big(T^*(m)\big) &= L_n^G\big(T^*(m)\big) + P_n^G\big(T^*(m)\big) \\
&= L_n^G(T^*(m)) + C_n
\end{aligned} \tag{3.51}$$

for all $m \in \Theta$. On event E_1, we have $\hat{m}^{or} \in \Theta$. We thus similarly have

$$\begin{aligned}
Q_n^G\big(T^*(\hat{m}^{or})\big) &= L_n^G\big(T^*(\hat{m}^{or})\big) + P_n^G\big(T^*(\hat{m}^{or})\big) \\
&= L_n^G\big(T^*(\hat{m}^{or})\big) + C_n.
\end{aligned} \tag{3.52}$$

Since the penalty terms for m and $\hat{m}^{or}$ are equal, it remains to compare their goodness-of-fit terms. Recall that $\hat{\alpha}^{or}$ is the unique global minimizer of $L_n^G(\alpha)$. Then for all $\hat{\alpha}^{or} \neq T^*(m)$, we have

$$L_n^G(\hat{\alpha}^{or}) < L_n^G(T^*(m)). \tag{3.53}$$

Then on the event E_1, (3.51), (3.52), and (3.53) together imply

$$Q_n^G(\hat{\alpha}^{or}) < Q_n^G\big(T^*(m)\big) \tag{3.54}$$

for all $T^*(m) \neq \hat{\alpha}^{or}$. Now, by (3.46), we have

$$Q_n^G(\hat{\alpha}^{or}) = Q_n(\hat{m}^{or}) \quad \text{and} \quad Q_n^G\big(T^*(m)\big) = Q_n\Big(T^{-1}\big(T^*(m)\big)\Big) = Q_n(m^*). \tag{3.55}$$

Then by (3.54) and (3.55), on event E_1, we have $Q_n(\hat{m}^{or}) < Q_n(m^*)$ for all $m^* \neq \hat{m}^{or}$, which completes the proof of (i). Note that $P(E_1) \to 1$ as $n \to \infty$.

Proof of Step 2: For a positive sequence t_n, denote

$$\Theta_n = \{\, m : \|m - \hat{m}^{\text{or}}\| \leq t_n \,\}. \tag{3.56}$$

Recall from (3.44) and (3.45) that

$$\begin{aligned} Q_n(m) &= L_n(m) + P_n(m) \\ &= \sum_{i=1}^{n} h(y_i - m_i) + \lambda \sum_{i<j} \rho(|m_i - m_j|). \end{aligned}$$

Let $m \in \Theta_n \cap \Theta$. Consider a Taylor expansion of $Q_n(m)$ about m^*:

$$Q_n(m) = Q_n(m^*) + \Gamma_1 + \Gamma_2, \tag{3.57}$$

where

$$\begin{aligned} \Gamma_1 &= \sum_{i=1}^{n} \frac{\partial L_n(m^a)}{\partial m_i}(m_i - m_i^*) \\ \Gamma_2 &= \sum_{i=1}^{n} \frac{\partial P_n(m^a)}{\partial m_i}(m_i - m_i^*) \end{aligned} \tag{3.58}$$

and $m^a = d_0 m + (1 - d_0)m^*$ with $d_0 \in (0, 1)$. We address Γ_1 and Γ_2 separately and derive a lower bound on each.

Consider Γ_2. Recall from (3.44) that $P_n(m) = \lambda \sum_{i<j} \rho(|m_i - m_j|)$ where $\rho(t) = \lambda^{-1} p_\omega(t, \lambda)$,

and denote $\bar{\rho}(t) = \text{sign}(t)\rho'(|t|)$. Then

$$\begin{aligned}
\Gamma_2 &= \sum_{i=1}^{n} \frac{\partial P_n(m^a)}{\partial m_i}(m_i - m_i^*) && \text{(going from (3.58))} \\
&= \lambda \sum_{i<j} \bar{\rho}(m_i^a - m_j^a)(m_i - m_i^*) + \lambda \sum_{j<i} \bar{\rho}(m_i^a - m_j^a)(m_i - m_i^*) && \text{(calculate derivative)} \\
&= \lambda \sum_{i<j} \bar{\rho}(m_i^a - m_j^a)(m_i - m_i^*) + \lambda \sum_{i<j} \bar{\rho}(m_j^a - m_i^a)(m_j - m_j^*) && \text{(switch labels)} \\
&= \lambda \sum_{i<j} \bar{\rho}(m_i^a - m_j^a)(m_i - m_i^*) - \lambda \sum_{i<j} \bar{\rho}(m_i^a - m_j^a)(m_j - m_j^*) && \text{(since } \bar{\rho}(t) = -\bar{\rho}(-t)\text{)} \\
&= \lambda \sum_{i<j} \bar{\rho}(m_i^a - m_j^a)\big[(m_i - m_i^*) - (m_j - m_j^*)\big] && (3.59)
\end{aligned}$$

For an example calculation of the derivative in the second line, see Appendix A.1. Now when $i, j \in G_k$, we have $m_i^* = m_j^*$, and so

$$\begin{aligned}
m_i^a - m_j^a &= d_0 m_i + (1 - d_0)m_i^* - \big[d_0 m_j + (1 - d_0)m_j^*\big] \\
&= d_0(m_i - m_j) + (1 - d_0)(m_i^* - m_j^*) && \text{(re-arrange terms)} \\
&= d_0(m_i - m_j) && \text{(since } m_i^* - m_j^* = 0 \text{ when } i, j \in G_k\text{)}
\end{aligned}$$

Thus, $m_i^a - m_j^a$ has the same sign as $m_i - m_j$ since $d_0 \in (0, 1)$. Then whenever $i, j \in G_k$, we have

$$\begin{aligned}
&\bar{\rho}(m_i^a - m_j^a)\big[(m_i - m_i^*) - (m_j - m_j^*)\big] && \text{(from (3.59))} \\
&\quad = \bar{\rho}(m_i^a - m_j^a)(m_i - m_j) && \text{(since } m_i^* - m_j^* = 0 \text{ when } i, j \in G_k\text{)} \\
&\quad = \rho'(|m_i^a - m_j^a|)\ \text{sign}(m_i^a - m_j^a)(m_i - m_j) && \text{(by definition of } \bar{\rho}\text{)} \\
&\quad = \rho'(|m_i^a - m_j^a|)|m_i - m_j|. && \text{(since sign}(m_i^a - m_j^a) = \text{sign}(m_i - m_j)\text{)}
\end{aligned}$$

Using this result and continuing from (3.59), we have

$$\begin{aligned}\Gamma_2 = \lambda \sum_{k=1}^{K} \sum_{\{i,j \in G_k, i<j\}} \rho'(|m_i^a - m_j^a|)|m_i - m_j| \\ + \lambda \sum_{k<k'} \sum_{\{i \in G_k, j \in G_{k'}\}} \bar{\rho}(m_i^a - m_j^a)[(m_i - m_i^*) - (m_j - m_j^*)].\end{aligned} \tag{3.60}$$

Now, note that

$$\begin{aligned}\left\|m^* - m^0\right\|_\infty &= \left\|\alpha - \alpha^0\right\|_\infty && (T(m^*) = \alpha \text{ and } T(m^0) = \alpha^0) \\ &\leq \left\|m - m^0\right\|_\infty && (3.61)\end{aligned}$$

where the inequality follows from (3.49). Then we also have

$$\begin{aligned}\left\|m^a - m^0\right\|_\infty &= \left\|d_0 m + (1-d_0)m^* - m^0\right\|_\infty && (\text{plug in } m^a) \\ &= \left\|d_0 m + m^* - d_0 m^* - m^0\right\|_\infty && (\text{expand out terms}) \\ &\leq \left\|m^* - m^0\right\|_\infty + \left\|d_0 m - d_0 m^0\right\|_\infty && (\text{triangle inequality}) \\ &\leq \left\|m^* - m^0\right\|_\infty \\ &\leq \left\|m - m^0\right\|_\infty && (\text{follows from (3.61)}) \\ \implies -2\left\|m^a - m^0\right\|_\infty &\geq -2\left\|m - m^0\right\|_\infty && (3.62)\end{aligned}$$

Now, for $k \neq k', i \in G_k, j \in G_{k'}$, note that

$$\begin{aligned}|m_i^0 - m_j^0| &= |m_i^0 - m_i^a + m_i^a - m_j^a + m_j^a - m_j^0| && (\text{add zero twice}) \\ &\leq |m_i^a - m_j^a| + |m_i^a - m_i^0| + |m_j^a - m_j^0| && (\text{triangle inequality}) \\ &\leq |m_i^a - m_j^a| + 2\left\|m^a - m^0\right\|_\infty && (\text{twice max is} \geq \text{than any other two summed})\end{aligned}$$

Re-arranging this result, we have

$$
\begin{aligned}
|m_i^a - m_j^a| &\geq |m_i^0 - m_j^0| - 2\left\|m^a - m^0\right\|_\infty && \text{(re-arranging the above)} \\
&\geq \min_{i \in G_k, j \in G_{k'}} |m_i^0 - m_j^0| - 2\left\|m^a - m^0\right\|_\infty \\
&= b_n - 2\left\|m^a - m^0\right\|_\infty && \text{(follows from definition of } b_n\text{)} \\
&\geq b_n - 2\left\|m - m^0\right\|_\infty && \text{(by (3.62))} \\
&\geq b_n - 2g_n && \text{(since } m \in \Theta \cap \Theta_n\text{)} \\
&\geq b_n && \text{(since } g_n \geq 0\text{)} \\
&> \kappa\lambda && (b_n > \kappa\lambda \text{ by Theorem 3 assumption})
\end{aligned}
$$

Then by (C5), $\rho(t)$ is constant for all $t \geq \kappa\lambda$, so then $\bar{\rho}(m_i^a - m_j^a) = 0$ for $i \in G_k, j \in G_{k'}, k \neq k'$. Thus Γ_2 simplifies from (3.60) to

$$
\Gamma_2 = \lambda \sum_{k=1}^{K} \sum_{\{i,j \in G_k, i<j\}} \rho'(|m_i^a - m_j^a|)|m_i - m_j| \tag{3.63}
$$

Notice that

$$
\left\|m^* - \hat{m}^{or}\right\|_\infty = \left\|\alpha - \hat{\alpha}^{or}\right\|_\infty \leq \left\|m - \hat{m}^{or}\right\|_\infty \tag{3.64}
$$

where the inequality follows from the same reasoning as that which led to (3.49). Then for $i, j \in G_k$ with $i < j$, we have

$$
\begin{aligned}
|m_i^a - m_j^a| &= |m_i^a - m_i^* + m_i^* - m_j^* + m_j^* - m_j^a| && \text{(add zero twice)} \\
&\leq |m_i^a - m_i^*| + |m_j^a - m_j^*| + |m_i^* - m_j^*| && \text{(triangle inequality)} \\
&= |m_i^a - m_i^*| + |m_j^a - m_j^*| && (m_i^* - m_j^* = 0 \text{ when } i, j \in G_k)
\end{aligned}
$$

$$
\begin{aligned}
&\leq 2\big\|m^a - m^*\big\|_\infty && \text{(twice max is } \geq \text{ than any adding other two)}\\
&= 2\big\|d_0 m + (1-d_0)m^* - m^*\big\|_\infty && \text{(plug in } m^a\text{)}\\
&= 2\big\|d_0 m - d_0 m^*\big\|_\infty && \text{(simplify)}\\
&= 2d_0\big\|m - m^*\big\|_\infty && \text{(pull out } d_0\text{)}\\
&\leq 2\big\|m - m^*\big\|_\infty && \text{(since } d_0 \in (0,1)\text{)}\\
&= 2\big\|m - \hat{m}^{or} + \hat{m}^{or} - m^*\big\|_\infty && \text{(add zero)}\\
&\leq 2\Big(\big\|m - \hat{m}^{or}\big\|_\infty + \big\|m^* - \hat{m}^{or}\big\|_\infty\Big) && \text{(Triangle Inequality)}\\
&\leq 4\big\|m - \hat{m}^{or}\big\|_\infty && \text{(follows from (3.64))}\\
&\leq 4t_n && \text{(because } m \in \Theta \cap \Theta_n\text{)}
\end{aligned}
$$

Hence, $\rho'(|m_i^a - m_j^a|) \geq \rho'(4t_n)$ by concavity of $\rho(\cdot)$, so then from (3.63), we have the bound

$$\Gamma_2 \geq \lambda \sum_{k=1}^{K} \sum_{\{i,j \in G_k,\, i<j\}} \rho'(4t_n)|m_i - m_j|. \tag{3.65}$$

Now consider Γ_1. Recall from (3.44) that $L_n(m) = \sum_{i=1}^n h(y_i - m_i)$. Note that

$$\frac{\partial L_n(m^a)}{\partial m_i} = -h'(y_i - m_i^a).$$

Let $q_i = h'(y_i - m_i^a) = \phi(y_i - m_i^a)$ and $q = (q_1, q_2, \ldots, q_n)$. Then

$$
\begin{aligned}
\Gamma_1 &= \sum_{i=1}^{n} \frac{\partial L_n(m^a)}{\partial m_i}(m_i - m_i^*) && \text{(going from (3.58))}\\
&= -\sum_{i=1}^{n} q_i(m_i - m_i^*) && \text{(using the above notation)}\\
&= -\sum_{k=1}^{K} \sum_{\{i,j \in G_k\}} \frac{q_i(m_i - m_j)}{n_k}
\end{aligned}
$$

$$
\begin{aligned}
&= -\sum_{k=1}^{K} \sum_{\{i,j \in G_k\}} \frac{(q_j - q_i)(m_j - m_i)}{2n_k} \\
&= -\sum_{k=1}^{K} \sum_{\{i,j \in G_k, i<j\}} \frac{(q_j - q_i)(m_j - m_i)}{n_k} \\
&\geq -\sum_{k=1}^{K} \sum_{\{i,j \in G_k, i<j\}} \frac{\max_{i,j} \{|q_j - q_i|\} \times |m_i - m_j|}{n_{\min}}. \qquad (3.66)
\end{aligned}
$$

where the results in the last several lines can be verified by straightforward calculation (see Appendix A.2). Denote $s_i = y_i - m_i^a$ and $s = y - m^a$. Since $y = m^0 + \epsilon$, we have

$$
s = \epsilon + m^0 - m^a. \qquad (3.67)
$$

Now,

$$
\begin{aligned}
\max_{i,j} |q_j - q_i| &\leq \max_{i,j} \{|q_j| + |q_i|\} && \text{(triangle inequality)} \\
&\leq 2\|\mathbf{q}\|_\infty && \text{(twice the max is } \geq \text{ than sum of any other two)} \\
&\leq 2\|s\|_\infty && \text{(since each } |q_i| \leq |s_i| \text{ by (D4))} \\
&\leq 2\|\epsilon\|_\infty + 2\|m^0 - m^a\|_\infty && \text{(plug in (3.67) and use triangle inequality)} \\
&\leq 2\|\epsilon\|_\infty + 2\|m^0 - m\|_\infty && \text{(from (3.62))} \\
&\leq 2\|\epsilon\|_\infty + 2g_n. && \text{(since } m \in \Theta \cap \Theta_n\text{)}
\end{aligned}
$$

Now, by Markov's Inequality, we have

$$
\begin{aligned}
P\big(|e_i| \geq n^d\big) = P\big(|e_i|^\zeta \geq n^{\zeta d}\big) &\leq \frac{E\big(|e_i|^\zeta\big)}{n^{\zeta d}} \\
\implies P\big(|e_i| < n^{\zeta d}\big) &\geq 1 - \frac{E\big(|e_1|^\zeta\big)}{n^{\zeta d}}. \qquad (3.68)
\end{aligned}
$$

Then

$$\begin{aligned} P\big(2\|e\|_\infty \le 2n^d\big) &= P\big(\|e\|_\infty \le n^d\big) \\ &= \prod_{i=1}^{n} P\big(|e_i| \le n^d\big) && \text{(by independence (D6))} \\ &= \prod_{i=1}^{n} P\big(|e_i|^\zeta \le n^{\zeta d}\big) \\ &\ge \left(1 - \frac{E\big(|e_1|^\zeta\big)}{n^{\zeta d}}\right)^n. && \text{(by (3.68))} \end{aligned}$$

Since $E\big(|e_1|^\zeta\big)$ is finite and $\zeta d > 1$ by theorem assumption, then $\left(1 - \frac{E\big(|e_1|^\zeta\big)}{n^{\zeta d}}\right)^n \to 1$ as $n \to \infty$. Thus, there is an event E_2 with $P(E_2) \ge \left(1 - \frac{E\big(|e_1|^\zeta\big)}{n^{\zeta d}}\right)^n$, and on E_2 we have

$$\begin{aligned} \max_{i,j} |q_j - q_i| &\le 2\|\epsilon\|_\infty + 2g_n && \text{(rewriting from above)} \\ &\le 2n^d + 2g_n. && (3.69) \end{aligned}$$

Now, note that

$$\begin{aligned} \frac{n_{\min}^{-1}(2n^d + 2g_n)}{g_n} &= \frac{2n^d}{n_{\min}\, g_n} + \frac{2n_{\min}^{-1}\, g_n}{g_n} && \text{(expand)} \\ &= \frac{2n^d}{|\gamma|^{-1}n^a + n_{\min}\|r_n\|_\infty} + 2n_{\min}^{-1} && \text{(plug in } g_n \text{ and simplify)} \\ &\to 0 \quad \text{as} \quad n \to \infty. && \text{(since } d < a \text{ by theorem assumption)} \end{aligned}$$

In particular, this shows that $n_{\min}^{-1}(2n^d + 2g_n) \ll g_n$. Hence,

$$\begin{aligned} n_{\min}^{-1} \max_{i,j} |q_j - q_i| &\le n_{\min}^{-1}(2n^d + 2g_n) && \text{(from (3.69))} \\ &\ll g_n && \text{(from above)} \\ &\ll \lambda, && (3.70) \end{aligned}$$

where the last line follows by theorem assumption.

Let $t_n \to 0$ so that by (D8), we have $\rho'(4t_n) \to 1$. Then by (3.65), (3.66), and (3.70), we have

$$\begin{aligned}\Gamma_1 + \Gamma_2 &\geq \sum_{k=1}^{K} \sum_{\{i,j \in G_k, i<j\}} \left(\lambda \rho'(4t_n) - \frac{\max_{i,j} \{|q_j - q_i|\}}{n_{\min}} \right) |m_i - m_j| \\ &\geq 0\end{aligned}$$

for large enough n. Thus, (ii) is proved, since $Q_n(m) - Q_n(m^*) = \Gamma_1 + \Gamma_2 \geq 0$. ■

3.5 Simulation Studies and Real Data Performance

We conduct some simulation studies in order to demonstrate that solution path clustering with robust loss, as opposed to the usual least squares loss, improves the search for clusters. We also apply the methods to a real data set. We evaluate performance with a few different criteria. When the true cluster location can be specified (e.g. we know the model which generated the data), we calculate the mean squared error and average absolute deviation. If there are K clusters with true cluster locations α_j, $j = 1, 2, \ldots, K$, and if the clustering method is able to find a K-component solution with cluster location estimates $\hat{\alpha}_j$, then we calculate

$$\text{MSE} = \frac{1}{K} \sum_{j=1}^{K} (\alpha_j - \hat{\alpha}_j)^2 \quad \text{and} \quad \text{AAD} = \frac{1}{K} \sum_{j=1}^{K} |\alpha_j - \hat{\alpha}_j|.$$

These criteria help us measure the biasedness of the estimates. Recall that the robust loss and the concave penalty are motivated by reducing the bias in the search for clusters. Hence, it is appropriate to compare the robust loss solutions with the least squares loss solutions

to explore if the robust estimates prove to be less biased. From a clustering point of view, evaluating these criteria may seem unimportant since the cluster location estimates are not as important as the estimated cluster partition. However, they are useful measurements because we hypothesize that bias reduction aids in the overall search for clusters throughout the entire solution path. Thus, evaluating the estimated cluster locations against the true locations provides some window into assessing the level of bias reduction.

To evaluate the cluster accuracy – that is, the ability to recover the true cluster partition – we use the popular Rand Index (Rand, 1971). The Rand Index essentially calculates the level of agreement between the estimated cluster partition and the true cluster partition. The Rand Index lies between 0 and 1, where a Rand Index of 0 indicates that they do not agree anywhere and 1 indicates that they are exactly the same. For a sample of size n, a clustering method forms a partition of the indices $\{1, 2, \ldots, n\}$. Denote the estimated cluster partition as $G = \left\{G_1, G_2, \ldots, G_{\hat{K}}\right\}$ and the true cluster partition as $T = \left\{T_1, T_2, \ldots, T_K\right\}$. Let TP be the number of true positive decisions; that is, the number of pairs of elements from the same true group are in the same estimated group. Let TN be the number of true negative decisions; that is, the number of pairs of elements from different true groups are in different estimated groups. Thus, $TP + TN$ can be regarded as a measure of agreement between the true cluster partition and the estimated cluster partition. Then we divide this number by the total number of pairwise decisions (adding up all true positives, true negatives, false positives, and false negatives) to obtain the Rand Index formula:

$$\text{Rand Index} = \frac{TP + TN}{\binom{n}{2}}. \tag{3.71}$$

The higher the Rand Index, the better that the clustering method recovers the true clustering structure.

Note that when there is any overlap between clusters, it is essentially impossible to recover the true clustering structure. This is especially the case when there are outliers present. Of course, one could regard outliers as not belonging to any cluster. In any case, it is questionable whether evaluating the agreement between a true partition and estimated partition is scientifically useful, since one of the goals of unsupervised learning is to discover data structures that were not previously known. For more discussion on this issue, see Section 4.3.5 below. Nevertheless, it is still a useful criteria to validate the performance of a clustering method since we know what is "should" be recovering.

In solution path clustering, it is possible that it is unable to find a K-cluster solution for a given sequence of λ values. If the solution path does not contain a K-cluster solution, this can be regarded as a weakness of the method. We could perform an exhaustive search if a particular solution path cannot find a K-cluster solution, but for the sake of computational considerations, we do not. We instead report the "viability rate," defined to be the number of datasets out of the total number of replicates that the solution path is able to find a K-cluster solution. Note that we construct the solution paths using a step-size of 0.01 or 0.02, which seems to be a fairly fine grid. For example, for several solution paths which did not find a K-component solution, we tried many λ values (even going out to the fourth or fifth decimal place) between the $(K-1)$-component solution and the $(K+1)$-component solution and still could not find a λ which gave a K-component solution. While it is certainly theoretically possible that a K-component solution really

does exist, it seems that other solutions are better local minimums to the objective function than whatever K-component solution might exist. Note that the MSE, AAD, and Rand Index are calculated *only* for the solutions for which the correct number of clusters are found. Thus, the viability rate becomes an important criterion if the other criteria between two methods are roughly similar the their viability rates are very different.

We also include some preliminary results on choosing the number of clusters via the modified BIC (Wang, Li, and Leng, 2009). The solution path clustering scheme produces a tree-like structure of varying numbers of clusters between 1 and n, similar to hierarchical clustering and dendrograms. While the entire solution path is of interest to the researcher for exploring how clusters are formed, there may be a need to automatically select the number of clusters. For example, we may want a data-driven justifcation for the number of clusters instead of resorting to a domain-specific explanation for the number of clusters. There is a large literature on selecting the number of clusters (e.g. Mirkin, 2011, and Xu et. al., 2016). The problem is difficult because it is not easy to define exactly what a correct clustering should be. Clustering can be very field-specific and require expert knowledge in order to be interpreted properly. It is also seems a bit strange to define a "correct" number of clusters since one of the whole points of clustering, as unsupervised learning, is to discover new structures in the data. On the other hand, when we can plot the data, or when we generate the data ourselves, it can be natural to define what the correct number of clusters should be.

The modified BIC can be written as

$$\text{BIC} = -\frac{2}{n}\ell_n(\hat{m}) + C_n\frac{\log n}{n}\hat{K}, \tag{3.72}$$

where $\ell_n(\hat{m})$ denotes the log-likelihood evaluated at the solution $\hat{m}(\lambda)$, $\hat{K}$ denotes the estimated number of clusters (that is, the number of unique values in $\hat{m}$), and C_n is a positive sequence possibly diverging to ∞. For example, C_n can equal $c \log n$ or $c \log \log n$ for some constant $c > 0$. If $C_n = 1$, then (3.72) simplifies to the traditional BIC (Schwarz, 1978). We select the model which corresponds to the minimum value of (3.72); that is, from a given solution path, we select the model that corresponds to

$$\lambda_{\text{BIC}} = \underset{\lambda}{\text{argmin}}\, \text{BIC}(\lambda) = -\frac{2}{n}\ell_n\big(\hat{m}(\lambda)\big) + c \log \log n \frac{\log n}{n}\hat{K}(\lambda),$$

where the solution $\hat{m}(\lambda)$ depends on λ. We use $C_n = c \log \log n$ and report results for $c = 5, 10, 15$. Ideally, we would like to see that the BIC performs well for a wide range of c values and is not too sensitive to the choice of c. Note that the BIC is very fast to calculate compared to other model selection methods that require cross-validation type techniques or bootstrap sampling.

The BIC can be interpreted as a goodness-of-fit term which measures model adequacy, plus a penalty term to encourage parsimony. If the penalty term is too strong, then the chosen model may tend to underfit. If the penalty term is too weak, the chosen model may tend to overfit. The C_n term modifies the traditional BIC (in which $C_n = 1$) to control the effect of a growing number of parameters along with the sample size. The traditional BIC assumes that the number of parameters is fixed as the sample size grows, which does not seem to be in keeping with real practice. One would expect that as an analysis accumulates a larger and larger sample size, more parameters would be included. Thus, a modified BIC to accommodate a growing number of parameters seems appropriate.

The BIC is often used for model selection when each of the candidate models

comes from the same family of distributions. For example, it is common to see the BIC in a regression settings where each candidate model assumes Gaussian errors. In these cases, any common integration constants of the likelihood function can be omitted from the BIC calculation since they will be the same across all evaluations. In our studies, it is of primary interest to compare solution path clustering with robust loss versus least squares loss. This means that the likelihoods will differ, and we must include the full likelihood (integration constants and all) in the BIC calculation in order to keep the different models comparable.

In the familiar least squares setting, we have

$$-\frac{2}{n}\ell_n(\hat{m}) = \log(\hat{\sigma}^2) + \log(2\pi) + 1,$$

where $\hat{\sigma}^2 = n^{-1}\sum_{i=1}^{n}(y_i - \hat{m}_i)^2$. Note that when the common integration constants are omitted, it reduces to just $\log\hat{\sigma}^2$, which might look more familiar.

3.5.1 Deriving the Huber Density

The robust loss function that we focus on the most is the Huber approximation to the absolute loss. Thus, we must derive $-2n^{-1}l_n(\hat{m})$ when the errors terms are assumed to follow a probability distribution in which the Huber loss function arises from maximum likelihood estimation. This amounts to deriving the "Huber density." Recall that the Huber loss function can be expressed as

$$h(x) = \begin{cases} \dfrac{x^2}{2r} & \text{if} \quad x \in [-r, r] \\ |x| - \dfrac{r}{2} & \text{if} \quad x \notin [-r, r] \end{cases} \tag{3.73}$$

for some given threshold parameter $r > 0$ (note that Huber loss is often expressed as the above multiplied by r. However, in the form (3.73), the function value more closely

approximates the absolute loss for small r). Thus, it is the least squares loss in a region centered at 0, and outside that region, it becomes the absolute loss. In other words, it is as if the Huber loss function is the result of a model that is comprised of a Gaussian distribution truncated to $[-r, r]$ and a Laplace distribution truncated to $[-r, r]^C$ (the superscript C here means "complement"). We can hence express the "Huber density" as

$$f(x) \propto \exp\big[-h(x)\big] = \exp\Big(-\frac{x^2}{2r}\Big) I\big(x \in [-r, r]\big) + \exp\Big(-|x| + \frac{r}{2}\Big) I\big(x \in [-r, r]^C\big),$$

where $I(\cdot)$ is the indicator function. To find the integration constant, we need to calculate

$$\int_{-\infty}^{\infty} \exp\big[-h(x)\big]\,dx = \int_{-r}^{r} \Big(-\frac{x^2}{2r}\Big)dx + 2\int_{r}^{\infty} \exp\Big(-|x| + \frac{r}{2}\Big)dx.$$

For the first term, we have

$$\int_{-r}^{r} \exp\Big(-\frac{x^2}{2r}\Big)dx = \sqrt{2\pi r}\big[\Phi\big(\sqrt{r}\big) - \Phi\big(-\sqrt{r}\big)\big],$$

where $\Phi(\cdot)$ is the cumulative distribution function of the standard normal. For the second term, we have

$$\begin{aligned} 2\int_{r}^{\infty} \exp\Big(-|x| + \frac{r}{2}\Big)dx &= 2\exp(r/2)\int_{r}^{\infty} \exp(-x)dx \\ &= 2\exp\Big(-\frac{r}{2}\Big). \end{aligned}$$

Then the full Huber density is

$$f(x; r) = C(r)\exp\big[-h(x)\big],$$

where

$$C(r) = \sqrt{2\pi r}\big[\Phi\big(\sqrt{r}\big) - \Phi\big(-\sqrt{r}\big)\big] + 2\exp\Big(-\frac{r}{2}\Big).$$

There is no analogous derivation for the Tukey loss function. Hence, we cannot be sure that the BIC results for the Tukey loss remain comparable against the other BIC results. We still include them, however, and for use in the BIC calculation, define

$$-\frac{2}{n}\ell_n(\hat{m}) := \frac{2}{n}\sum_{i=1}^{n} h(y_i - \hat{m}_i, r),$$

where h here is defined as in (3.30). Note that a similar form of Tukey BIC was also used in Chang, Robert, and Welsh (2018).

When using the Tukey and Huber loss functions, an estimate of the scale $\hat{\sigma}$ is usually required, so that the chosen threshold parameter can be $r = c\hat{\sigma}$ where $c > 0$ is some constant chosen to achieve a certain efficiency in the Gaussian error case (see Section 3.3). To estimate the scale, we first perform solution path clustering with the Huber approximation to the absolute loss. Then we use $\hat{\sigma} = \text{MAR}$ the median absolute residual evaluated at the true number of clusters. If the solution path did not have a solution at the true number of clusters, we used the mean of all the MAR's of the replicates which did obtain a correct solution. We then use $r = 1.345\hat{\sigma}$ and $r = 4.695\hat{\sigma}$ for the Huber and Tukey solution paths, respectively. We show these results to demonstrate that the Huber and Tukey functions can work (or not) with a good estimate of scale.

3.5.2 Simulation Study 1

Our first simulation study contains some straightforward settings with varying levels of outlier presence to evaluate the solution path clustering performances using Huber approximation to absolute loss, Huber loss, Tukey loss, and least squares loss. Let $N(\mu, v)$ represent the normal distribution with mean μ and variance v, and let T_g be the

t-distribution with g degrees of freedom. Let e_i represent the error terms, which are independent. We generated data according to the following models:

1. *(Normal errors)*: $P(m_i = 0) = P(m_i = 4) = \frac{1}{2}$ with $e_i \sim N(0, 1)$.
2. *(5% Contamination)*: $P(m_i = 0) = P(m_i = 4) = \frac{1}{2}$ with $e_i \sim 0.95N(0, 1) + 0.05N(0, 100)$.
3. *(10% Contamination)*: $P(m_i = 0) = P(m_i = 4) = \frac{1}{2}$ with $e_i \sim 0.90N(0, 1) + 0.10N(0, 100)$.
4. *(15% Contamination)*: $P(m_i = 0) = P(m_i = 4) = \frac{1}{2}$ with $e_i \sim 0.85N(0, 1) + 0.15N(0, 100)$.
5. *(t_2 errors)*: $P(m_i = 0) = P(m_i = 4) = \frac{1}{2}$ with $e_i \sim t_2$.
6. *(Normal errors)*: $P(m_i = 0) = P(m_i = 4) = P(m_i = 8) = P(m_i = 12) = \frac{1}{4}$ with $e_i \sim N(0, 1)$.
7. *(5% Contamination)*: $P(m_i = 0) = P(m_i = 4) = P(m_i = 8) = P(m_i = 12) = \frac{1}{4}$ with $e_i \sim 0.95N(0, 1) + 0.05N(0, 100)$.
8. *(10% Contamination)*: $P(m_i = 0) = P(m_i = 4) = P(m_i = 8) = P(m_i = 12) = \frac{1}{4}$ with $e_i \sim 0.90N(0, 1) + 0.10N(0, 100)$.
9. *(15% Contamination)*: $P(m_i = 0) = P(m_i = 4) = P(m_i = 8) = P(m_i = 12) = \frac{1}{4}$ with $e_i \sim 0.85N(0, 1) + 0.15N(0, 100)$.
10. *(t_2 errors)*: $P(m_i = 0) = P(m_i = 4) = P(m_i = 8) = P(m_i = 12) = \frac{1}{4}$ with $e_i \sim t_2$.

We did simulations using sample sizes $n = 40$ and $n = 200$, and generated 200 replicates. For $n = 40$, we simulated only from settings 1-5.

Tables 3.1, 3.2, and 3.3 contain the results for MSE, AAD, Rand, and Viability Rate averaged across 200 replicates. For the Rand calculations, we only used the observations which are *not* considered as outliers. For example, in the contamination cases, if the error is being generated from the large variance normal distribution, we do not include that observation in the Rand calculation. In the t_2 errors cases, we categorize an outlier with the following rule: if $e_i^2 \geq \chi^2_{0.95,\, df=1}$, then categorize the observation as an outlier. Else, it is not an outlier. Here, $\chi^2_{0.95,\, df=1} = 3.841459$ is the 95-th percentile of a Chi-squared distribution with 1 degree of freedom. This rule was used in Coretto and Hennig (2010), and originally comes from McLachlan and Peel (2000). During our simulation studies, the average rate of outlier occurrence for t_2 errors using this rule is approximately 19%. Hence, we simulate outliers with rates 0%, 5%, 10%, 15%, and $\approx$ 19% so that we can observe the performance measures across a spectrum of outlier presence.

In cases 2-5 and 7-10, that is, the cases where outliers are present, the robust loss function results are uniformly better across all of the criteria. In cases 1 and 6, the Normal cases, the least squares results are very slightly better, but the robust loss results are very comparable. This illustrates the principle that robust loss functions lose some efficiency in the Gaussian case. However, the slight decrease in performance in the Gaussian case seems a small price to pay for the large increase in performance for even a small preponderance of outliers. The BIC results are contained in Tables 3.4, 3.5, and 3.6. For the most part, there is a healthy range of c values where the correct number of components is chosen high

percentage of the time. Note that the least squares loss seems to be much more affected by the presence of outliers, which is reflected in its lower BIC performance for the outlier cases. The robust loss functions still perform quite well in the Normal errors cases.

Two components		MSE(α)	AAD(α)	Rand	Viability Rate
Normal errors	H-LAD	0.1061	0.2586	0.9384	0.99
	Huber	0.0781	0.2183	0.9373	0.995
	Tukey	0.0824	0.2258	0.9389	0.995
	LS	0.0782	0.2192	0.9371	1
5% Contamination	H-LAD	0.1204	0.2727	0.9296	1
	Huber	0.1091	0.2599	0.9296	1
	Tukey	0.1086	0.2574	0.9284	1
	LS	15.1889	1.3422	0.8764	0.845
10% Contamination	H-LAD	0.1412	0.2932	0.9248	0.995
	Huber	0.1168	0.2725	0.9339	0.985
	Tukey	0.1012	0.2523	0.9336	0.995
	LS	23.6792	1.9505	0.8597	0.7
15% Contamination	H-LAD	0.1607	0.316	0.916	0.98
	Huber	0.1696	0.3271	0.9282	0.965
	Tukey	0.1251	0.2748	0.9312	0.975
	LS	18.7301	1.8107	0.8763	0.585
t_2 errors	H-LAD	0.1786	0.316	0.941	0.98
	Huber	0.1648	0.3146	0.9482	0.975
	Tukey	0.1466	0.2895	0.9486	0.995
	LS	53.9089	1.7611	0.9067	0.875

Table 3.1: Solution path clustering performance results, averaged across 200 replicates, with various loss functions, corresponding to simulation settings 1-5 for $n = 40$. "H-LAD" means Huber approximation to least absolute deviation and "LS" means least squares.

Two components		MSE(α)	AAD(α)	Rand	Viability Rate
Normal errors	H-LAD	0.0316	0.1377	0.9384	1
	Huber	0.0245	0.1201	0.9356	1
	Tukey	0.0242	0.1198	0.9366	1
	LS	0.0312	0.1351	0.9347	1
5% Contamination	H-LAD	0.0345	0.1506	0.9354	1
	Huber	0.0303	0.1389	0.9308	1
	Tukey	0.0274	0.1296	0.9294	1
	LS	0.2002	0.3583	0.9263	1
10% Contamination	H-LAD	0.047	0.1741	0.9333	1
	Huber	0.0396	0.1623	0.9343	1
	Tukey	0.0267	0.1318	0.9357	1
	LS	0.4957	0.6113	0.9301	0.985
15% Contamination	H-LAD	0.0564	0.1939	0.9338	1
	Huber	0.0659	0.2151	0.9305	0.995
	Tukey	0.0304	0.1357	0.934	1
	LS	1.1074	0.9268	0.925	0.98
t_2 errors	H-LAD	0.0645	0.2035	0.9476	1
	Huber	0.0584	0.1876	0.953	1
	Tukey	0.0506	0.1766	0.9532	1
	LS	422.1142	2.3382	0.9301	0.95

Table 3.2: Solution path clustering performance results with various loss functions, corresponding to simulation settings 1-5 for $n = 200$. "H-LAD" means Huber approximation to least absolute deviation and "LS" means least squares.

Four components		MSE(α)	AAD(α)	Rand	Viability Rate
Normal errors	H-LAD	0.0696	0.2096	0.9528	1
	Huber	0.0469	0.1711	0.9515	1
	Tukey	0.0453	0.1679	0.9515	1
	LS	0.0591	0.1898	0.9508	1
5% Contamination	H-LAD	0.0888	0.2369	0.9497	1
	Huber	0.0652	0.201	0.9484	1
	Tukey	0.0583	0.1867	0.9489	1
	LS	7.4835	0.6877	0.9432	0.945
10% Contamination	H-LAD	0.1023	0.2536	0.9476	1
	Huber	0.0773	0.2198	0.9485	0.995
	Tukey	0.0592	0.1905	0.9498	1
	LS	6.0516	0.8129	0.9405	0.945
15% Contamination	H-LAD	0.1088	0.2627	0.9472	0.99
	Huber	0.1161	0.2612	0.9449	0.97
	Tukey	0.0701	0.2079	0.9461	1
	LS	6.8479	0.9711	0.9387	0.95
t_2 errors	H-LAD	0.1129	0.2669	0.9632	0.99
	Huber	0.1057	0.2489	0.9627	0.995
	Tukey	0.0946	0.2351	0.964	0.99
	LS	292.6375	2.7248	0.9363	0.885

Table 3.3: Solution path clustering performance results, averaged across 200 replicates, with various loss functions, corresponding to simulation settings 6-10 for $n = 200$. "H-LAD" means Huber approximation to least absolute deviation and "LS" means least squares.

Two components		c=5	c=10	c=15
Normal errors	H-LAD	0.885	0.995	1
	Huber	0.905	0.99	0.74
	Tukey	0.91	0.41	0.06
	LS	0.265	0.48	0.1
5% Contamination	H-LAD	0.805	0.99	0.995
	Huber	0.82	0.98	0.825
	Tukey	0.925	0.5	0.085
	LS	0.185	0.18	0.025
10% Contamination	H-LAD	0.665	0.955	0.98
	Huber	0.75	0.95	0.855
	Tukey	0.9	0.515	0.145
	LS	0.13	0.105	0.02
15% Contamination	H-LAD	0.65	0.975	0.965
	Huber	0.74	0.95	0.79
	Tukey	0.84	0.52	0.15
	LS	0.38	0.375	0.03
t_2 errors	H-LAD	0.935	0.99	0.98
	Huber	0.99	0.975	0.585
	Tukey	0.89	0.275	0.04
	LS	0.445	1	0.305

Table 3.4: BIC results for simulation settings 1-5 for $n = 40$ and 200 replicates. The c values are from $C_n = c \log\log n$ in the modified BIC. "H-LAD" means Huber approximation to least absolute deviation and "LS" means least squares.

Two components		c=5	c=10	c=15
Normal errors	H-LAD	0.035	0.87	0.995
	Huber	0.135	0.94	0.99
	Tukey	0.405	0.97	1
	LS	0	0.85	0.735
5% Contamination	H-LAD	0.025	0.805	0.98
	Huber	0.07	0.89	1
	Tukey	0.445	0.97	1
	LS	0.01	0.95	0.455
10% Contamination	H-LAD	0	0.57	0.975
	Huber	0.05	0.715	0.98
	Tukey	0.405	0.975	1
	LS	0.01	0.94	0.22
15% Contamination	H-LAD	0.005	0.6	0.935
	Huber	0.085	0.77	0.975
	Tukey	0.255	0.92	0.99
	LS	0	0.7	0.745
t_2 errors	H-LAD	0.075	0.94	0.99
	Huber	0.315	0.965	1
	Tukey	0.42	0.99	1
	LS	0	0.185	0.905

Table 3.5: BIC results for simulation settings 1-5 for $n = 200$ and 200 replicates. The c values are from $C_n = c \log\log n$ in the modified BIC. "H-LAD" means Huber approximation to least absolute deviation and "LS" means least squares.

Four components		c=5	c=10	c=15
Normal errors	H-LAD	0.57	0.985	0.995
	Huber	0.76	0.985	0.97
	Tukey	0.9	0.92	0.3
	LS	0.05	0.525	0.18
5% Contamination	H-LAD	0.46	0.99	1
	Huber	0.655	0.995	0.97
	Tukey	0.88	0.965	0.43
	LS	0.06	0.255	0.04
10% Contamination	H-LAD	0.29	0.95	0.985
	Huber	0.39	0.935	0.94
	Tukey	0.83	0.975	0.55
	LS	0.06	0.085	0
15% Contamination	H-LAD	0.33	0.965	0.99
	Huber	0.495	0.975	0.965
	Tukey	0.73	0.965	0.43
	LS	0.06	0.58	0.2
t_2 errors	H-LAD	0.765	1	1
	Huber	0.855	1	0.975
	Tukey	0.91	0.91	0.23
	LS	0.005	0.94	0.99

Table 3.6: BIC results for simulation settings 6-10 for $n = 200$ and 200 replicates. The c values are from $C_n = c \log\log n$ in the modified BIC. "H-LAD" means Huber approximation to least absolute deviation and "LS" means least squares.

3.5.3 Simulation Study 2

The simulation settings here follow those of Coretto and Hennig (2010). The substance of their article consisted of comparing robust clustering methods based on mixture models. They considered 6 different data generating schemes, trying "to cover a range of essentially different archetypical situations that are not obviously unrealistic" (Coretto and Hennig, page 120, 2010). The notation here is $N(\mu, v)$ for a normal distribution with mean μ and variance v, $U(a, b)$ is the uniform distribution on $[a, b]$, and $T_g(\mu, v)$ is the non-central t distribution with g degrees of freedom ($g > 2$), location parameter μ, and variance v.

1. *Side-noise* (3 or 4 clusters): $0.30N(0, 1.5)+0.25N(7, 2)+0.35N(14, 1.5)+0.10U(17, 25)$.

2. *Inside-noise* (3 or 4 clusters): $0.30N(0, 1.5)+0.25N(7, 1.5)+0.35N(21, 2)+0.10U(11, 19)$.

3. *Wide-noise* (2 clusters): $0.45N(7, 2) + 0.45N(14, 1.5) + 0.10U(0, 21)$.

4. *Outlier process* (2 clusters): $n-2$ points from $0.50N(0, 2)+0.50N(5, 1.2)$ and 2 points from $U(20, 25)$.

5. *t-noise* (3 clusters): $0.40T_3(0, 2) + 0.30T_{10}(6, 2) + 0.30T_{10}(12, 1)$.

6. *Gaussian mixture* (3 clusters): $0.40N(0, 2) + 0.30N(6, 2) + 0.30N(12, 1)$.

The λ sequence we used had a step size of 0.02. When the clusters are more well-separated, we can use a less fine grid of λ points.

Note that the goal of the methods compared in Coretto and Hennig (2010) is a bit different than ours. Their goal was so simultaneously discover clusters while also classifying noise. For example, some of the mixture models compared include a Uniform component

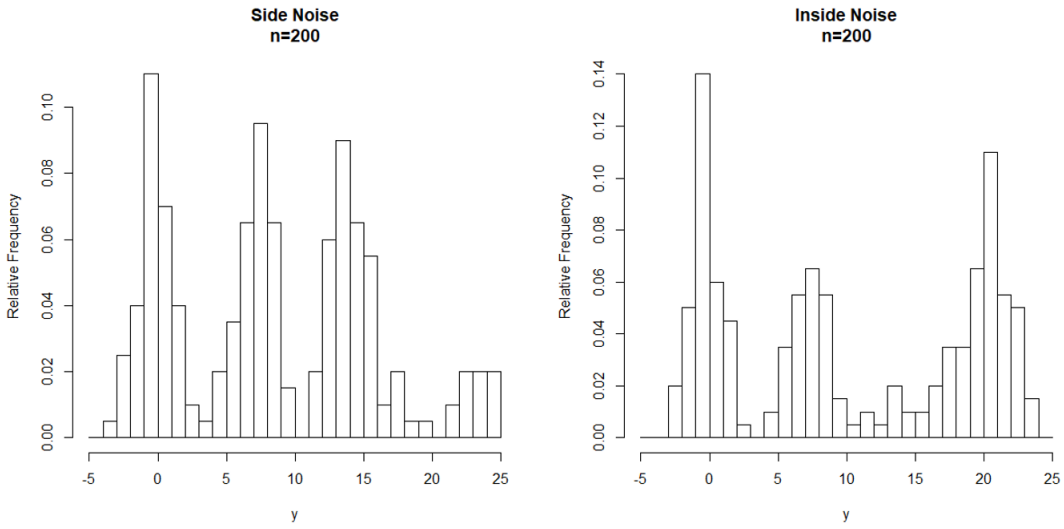


Figure 3.3: Histograms for Side Noise and Inside Noise with $n = 200$.

designed to capture the noise in the data. By contrast, the solution path clustering with robust loss is not designed to model or classify any noise. The robust loss functions were chosen mainly to mitigate undue influence from outliers or noise. In particular, solution path clustering will simply classify any noisy points into a cluster. Another aspect to note is that in the Side-noise and Inside-noise settings, it is unclear if there are really 3 or 4 clusters. In Tables 3.7 and 3.8, we calculate the criteria using 3 clusters and regard the uniform component as noise. Note that Coretto and Hennig (2010) adopt the same strategy by only comparing estimates with the true parameters for the non-noise components. Interestingly, when $n = 200$ for the Side-noise case, if we input the "true" number of clusters as four, than the viability rate drops almost as low as 55%. Thus, the solution path clustering seems to have much more trouble finding four cluster solutions than three cluster solutions for this case (and for the chosen grid of λ values), even though the histogram in Figure 3.3 seems to indicate that there may be a fourth cluster. On the other hand, the Inside-noise case

seems to only have three clusters.

The BIC results are contained in Tables 3.9 and 3.10. For the most part, all of the loss functions perform quite well. Even though most of the cases contain outliers and noises, the clusters are quite well-separated, so that even the least squares loss is performing decently. The Side Noise and Inside Noise cases deserve a closer look. When the sample size is $n = 50$, the BIC results are pretty good, meaning that the methods were able to recover three clusters are fairly high percentage of the time. However, when $n = 200$, the methods do not choose three clusters as often as when $n = 50$. Perhaps this it not too surprising since a larger overall sample size means that that Uniform "noise" component will begin to manifest as its own cluster as opposed to simply noise. This seems to be especially the case for the Side Noise, as evidenced by Figure 3.3. For this reason, we also include the proportion of the times that the BIC selected four components, shown in parentheses in Table 3.10. The Tukey loss performs quite poorly in the $n = 50$ case. At first we thought that this was because we do not have a full Tukey likelihood to use in the calculation, and so it might be directly comparable to the other likelihoods. However, trying c values from 1 to 30 did not show any improvements.

It is interesting to compare our methods to the mixture model methods used in Coretto and Hennig (2010). A few things should be noted before we begin comparisons. First, the simulation settings matched exactly the assumptions of at least one of their candidate mixture models. Second, the goal of fitting mixture models is much more ambitious than ours since they want to distinguish between noise and true clusters, while we do not. Third, for AAD(α), they only compute the upper trimmed 90% means, while we compute

the overall mean and do not trim. Fourth, our method might not even find a K component solution, while a mixture model always can if the EM algorithm converges. However, given that our robust loss functions' viability rates are almost always above 90%, we feel that these last two reasons are more or less balanced out, so that comparisons between our method and theirs are more or less fair.

For $n = 50$, our solution path clustering with robust loss uniformly outperforms all of the mixture models in Coretto and Hennig (2010) in terms of AAD(α). For $n = 200$, it is the same story although not as dramatic of a difference. Still, these comparisons should be taken with a grain of salt due to the reasons discussed in the previous paragraph. Nevertheless, the solution path clustering method's performance results for these simulation settings are encouraging.

		MSE(α)	AAD(α)	Rand	Viability Rate
Side Noise	H-LAD	4.0999	0.9497	0.9684	0.95
	Huber	4.2078	0.9356	0.9679	0.93
	Tukey	3.9371	0.8148	0.9699	0.925
	LS	4.7795	1.208	0.9637	0.93
Inside Noise	H-LAD	0.4803	0.4732	0.9961	0.975
	Huber	0.337	0.4115	0.9962	0.985
	Tukey	0.2755	0.3596	0.9964	0.98
	LS	0.7571	0.6168	0.9967	0.98
Wide Noise	H-LAD	0.1368	0.2987	0.9823	0.995
	Huber	0.1224	0.2784	0.9835	0.995
	Tukey	0.2316	0.373	0.9831	0.995
	LS	0.1666	0.3259	0.9827	1
Outlier Process	H-LAD	0.0865	0.2324	0.9511	1
	Huber	0.0772	0.2205	0.9446	1
	Tukey	0.1336	0.296	0.9597	1
	LS	144.1309	8.894	0.5663	0.34
T Noise	H-LAD	0.1688	0.3069	0.9977	0.985
	Huber	0.1111	0.2545	0.997	0.99
	Tukey	0.1723	0.3073	0.9964	0.99
	LS	0.1199	0.2632	0.9975	0.99
Gaussian	H-LAD	0.2136	0.3268	0.9825	0.99
	Huber	0.1279	0.2565	0.9828	0.99
	Tukey	0.1747	0.3051	0.9833	0.99
	LS	0.1137	0.2511	0.9846	0.995

Table 3.7: Solution path clustering performance results, averaged across 200 replicates, with various loss functions, corresponding to simulation settings 1-6 for $n = 50$. "H-LAD" means Huber approximation to least absolute deviation and "LS" means least squares.

		MSE(α)	AAD(α)	Rand	Viability Rate
Side Noise	H-LAD	1.7023	0.5199	0.9823	0.895
	Huber	1.2676	0.4534	0.9842	0.885
	Tukey	0.8595	0.2803	0.9856	0.91
	LS	1.9206	0.7476	0.9848	0.865
Inside Noise	H-LAD	0.1738	0.3106	0.9967	1
	Huber	0.1587	0.2933	0.9964	1
	Tukey	0.0511	0.1784	0.9961	1
	LS	0.5787	0.5308	0.9966	1
Wide Noise	H-LAD	0.061	0.1922	0.9869	1
	Huber	0.0358	0.1525	0.9848	1
	Tukey	0.0473	0.171	0.9851	1
	LS	0.0707	0.2214	0.9866	1
Outlier Process	H-LAD	0.0301	0.1376	0.964	1
	Huber	0.0217	0.1184	0.962	1
	Tukey	0.0299	0.1376	0.9615	1
	LS	0.0848	0.236	0.9593	1
T Noise	H-LAD	0.061	0.1893	0.9982	1
	Huber	0.0262	0.1259	0.9981	1
	Tukey	0.0332	0.1355	0.9989	1
	LS	0.0373	0.1493	0.9978	1
Gaussian	H-LAD	0.0562	0.1886	0.9872	1
	Huber	0.0247	0.1252	0.9873	1
	Tukey	0.0308	0.1396	0.9872	1
	LS	0.0235	0.1215	0.9873	1

Table 3.8: Solution path clustering performance results, averaged across 200 replicates, with various loss functions, corresponding to simulation settings 1-6 for $n = 200$. "H-LAD" means Huber approximation to least absolute deviation and "LS" means least squares.

		c=5	c=10	c=15
Side Noise	H-LAD	0.28	0.67	0.81
	Huber	0.45	0.73	0.725
	Tukey	0.56	0.08	0.005
	LS	0.26	0.155	0.005
Inside Noise	H-LAD	0.35	0.8	0.89
	Huber	0.555	0.83	0.895
	Tukey	0.6	0.07	0.02
	LS	0.44	0.56	0.035
Wide Noise	H-LAD	0.445	0.975	0.995
	Huber	0.935	0.995	0.995
	Tukey	0.135	0	0
	LS	0.575	1	0.825
Outlier Process	H-LAD	0.795	1	1
	Huber	0.985	0.965	0.51
	Tukey	0.025	0	0
	LS	0	0.005	0
t Noise	H-LAD	0.925	0.985	0.985
	Huber	0.98	0.855	0.315
	Tukey	0.055	0.005	0
	LS	0.815	0.945	0.245
Gaussian	H-LAD	0.945	0.985	0.985
	Huber	0.98	0.945	0.625
	Tukey	0.125	0	0
	LS	0.835	0.985	0.215

Table 3.9: BIC results for simulation settings 1-6 for $n = 50$ and 200 replicates. The c values are from $C_n = c \log\log n$ in the modified BIC. "H-LAD" means Huber approximation to least absolute deviation and "LS" means least squares.

		c=5	c=10	c=15
Side Noise	H-LAD	0 (0.05)	0.145 (0.375)	0.515 (0.30)
	Huber	0 (0.13)	0.25 (0.475)	0.56 (0.37)
	Tukey	0.34 (0.275)	0.83 (0.155)	0.79 (0.08)
	LS	0 (0.35)	0.18 (0.43)	0.51 (0.36)
Inside Noise	H-LAD	0 (0.08)	0.205 (0.535)	0.58 (0.33)
	Huber	0.02 (0.115)	0.36 (0.455)	0.65 (0.31)
	Tukey	0.2 (0.445)	0.925 (0.075)	0.895 (0.005)
	LS	0 (0.01)	0.345 (0.43)	0.67 (0.31)
Wide Noise	H-LAD	0	0.225	0.925
	Huber	0.04	0.93	1
	Tukey	0.575	0.885	0.29
	LS	0	0.71	1
Outlier Process	H-LAD	0.01	0.89	1
	Huber	0.65	1	1
	Tukey	0.91	0.115	0.005
	LS	0	0.215	0.78
t Noise	H-LAD	0.435	0.995	1
	Huber	0.875	1	1
	Tukey	0.98	0.33	0.01
	LS	0.01	0.925	0.99
Gaussian	H-LAD	0.14	1	1
	Huber	0.68	1	1
	Tukey	0.89	0.86	0.145
	LS	0	0.92	1

Table 3.10: BIC results for simulation settings 1-6 for $n = 200$ and 200 replicates. For the Side Noise and Inside, the number not in parentheses used three clusters as the "truth," and the number in parentheses used four clusters as the "truth." The c values are from $C_n = c \log\log n$ in the modified BIC. "H-LAD" means Huber approximation to least absolute deviation and "LS" means least squares.

3.5.4 President Dataset

Consider the President Dataset from Hayden (2005). It is a univariate dataset containing 43 observations corresponding to the number of days in office for 43 presidents of the United States. For a visual of the data, see the dotplot in Figure 3.4. Note that there are 24 presidents who fit the usual pattern of serving one or two terms (1461 days or 2922). The remaining 19 presidents do not fit this usual pattern of serving one or two terms. Hayden (2005) emphasizes that an outlier does not necessarily mean an extreme observation (although it could), but more broadly, that it means an observation that does not fit the pattern of the data. In this sense, the 19 presidents who did not serve the usual pattern of one or two terms could be considered as outliers, and which merit more investigation to explain why they do not fit in the norm. This means that 44% of this dataset can be considered as outliers.

We analyze the data with our solution path clustering method using least squares loss and the Huber approximation to absolute loss. Note that we first standardized the data to have mean 0 and standard deviation 1. Figure 3.5 contains the solution path plots. We compare the two cluster solutions from each path. Both solutions cluster 30 presidents in the first group and 13 in the second. Furthermore, both of the clustering partitions are identical, as can be evidenced by their solution path plots in Figure 3.5.

Let us now compare the cluster location estimates between the two solutions. 1461 days and 2922 days could be considered the "true" location estimates, corresponding to serving within the usual pattern of one or two terms. We might expect then that a good cluster location estimate would be close to these numbers. After transforming the estimates

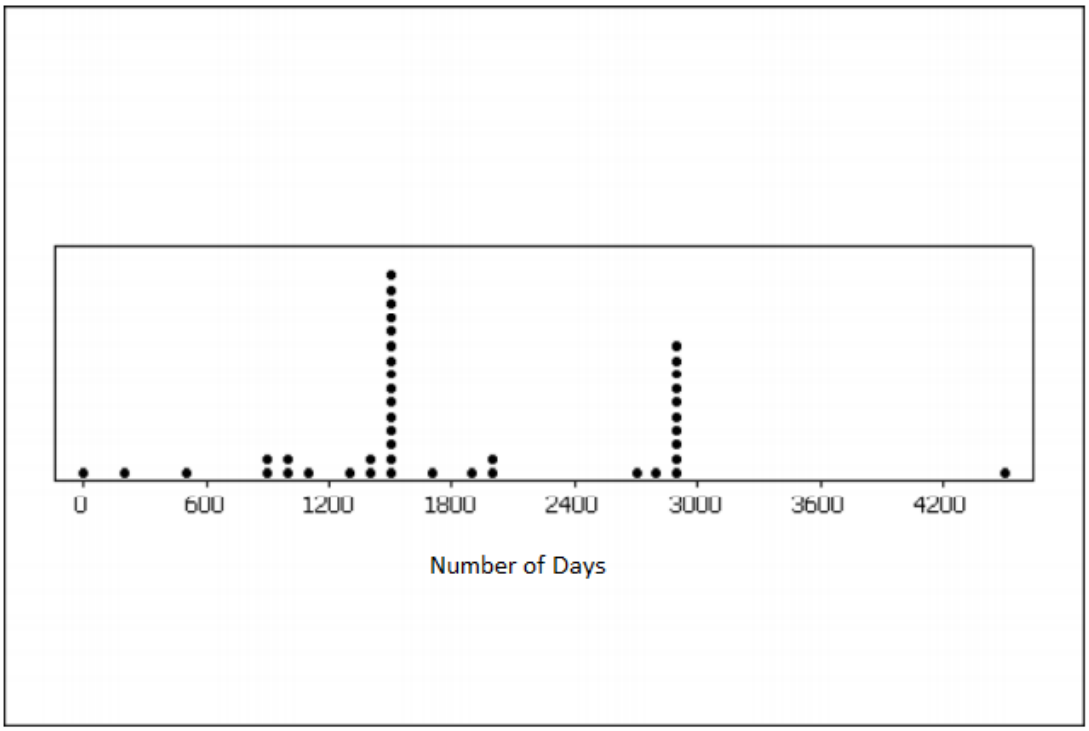


Figure 3.4: Dotplot of the number of days.

back to the original scale, the least squares solution path has cluster location estimates of 1308.9 and 3011.2. The Huber approximation to absolute loss solution path has cluster location estimates of 1418.0 and 2962.7. Hence, we see that using a robust loss function the solution path clustering scheme yields estimates that are closer to the "true" values of 1461 and 2922.

To be fair, however, from a strictly clustering perspective, the solutions are equivalent since they both form the same partition of the data. Normally, we do not need to use clustering methods on univariate datasets. If one wished to discover the group structure, there are many other methods that are more straightforward and avail themselves to easy visual checks, such as sorting the data, building a histogram or density plot, or even a

dotplot. However, since it is easy to know what can be considered outliers for this dataset, it is a good example to illustrate the principles of solution path clustering.

Out of curiosity, we also included the BIC results in Table 3.11. It is unfortunate to see that H-LAD did not select two clusters for our considered c values, since those c values performed decently in our simulation studies. Using LS also only had $c = 5$ able to select two clusters.

	c=5	c=10	c=15
H-LAD	1	1	1
LS	2	1	1

Table 3.11: Number of clusters chosen by the modified BIC with different c values for the President Data Set.

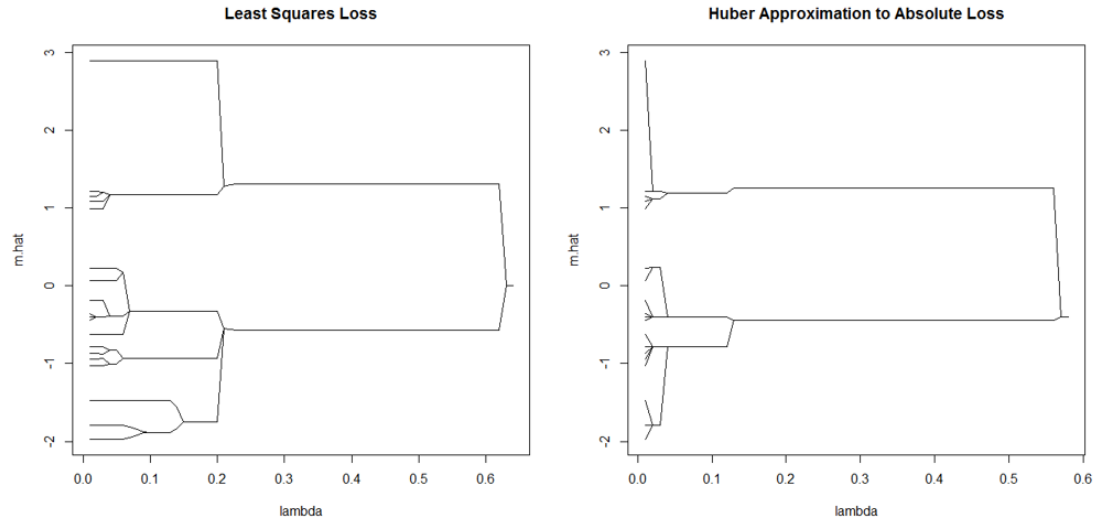


Figure 3.5: Solution path plots for the President Dataset. Note that the data was standardized before performing the analysis.

Chapter 4

Further Extensions

We now discuss some possible extensions to the solution path clustering method. The imperative is to extend to multivariate data. Learning the covariance structure and forming clusterings simultaneously appears to be a complicated matter. Without incorporating the covariance, the solution path clustering method implicitly assumes that $\Sigma = I$ the identity matrix, i.e. the data exhibit zero correlation and that each dimension has the same scale (a similar assumption made in the classic K-means method). Since we could not develop a sensible covariance learning strategy throughout the research that led to this dissertation, we instead explore the settings in which our method begins to fail. For example, how much correlation can be present in the data before our method fails to discover well-separated clusters? How does it handle differences in scale? What are the limits of assuming $\Sigma = I$? What are the limits of our method? These questions are explored in the simulation studies. The results in this chapter are far from conclusive, but hopefully they at least represent some tentative step forward. Reporting failed results does not seem to be

as common, but communicating what does *not* work is still a useful part of the scientific method.

4.1 Multivariate Data

Consider observations $y_1, y_2, \ldots, y_n$ of dimension d, where each y_i is associated with its own location vector m_i. Let M be the $n \times p$ matrix comprised of the m_i as row vectors. Let $\|a\| = \left(a^T a\right)^{1/2}$ be the Euclidean norm for any vector a. Then one might propose the objective function

$$Q_n(M; \lambda) = \frac{1}{2} \sum_{i=1}^{n} (y_i - m_i)^T \Sigma^{-1} (y_i - m_i) + \frac{n}{2} \log |\Sigma| + \sum_{i<j} p_\omega(\|m_i - m_j\|, \lambda). \qquad (4.1)$$

There are some key differences moving from the univariate case to the multivariate case. Firstly, the penalty function argument changed from the L_1 norm of the pairwise differences $(m_i - m_j)$ to the L_2 norm. This ensures that entire mean vector differences are shrunk towards zero. If we used the L_1 norm here, only component-wise differences would be shrunk since the L_1 norm separates across vector elements. Secondly, we are explicitly including the covariance parameter Σ. This is so the model can accommodate any correlation and scale differences among the dimensions. We may assume $\Sigma = I$ the identity matrix, which a similar assumption made in the classic K-means method, but this is likely an oversimplification with poorer performance capabilities (see the end of Section 1.1).

Note that this objective function can be developed from the model

$$Y_i \sim MVN(m_i, \Sigma)$$

for each i, all independent. Note that a common covariance Σ is assumed.

Developing the estimation procedure follows analogously from the univariate case (Section 3.2.2). After re-parametrizing the objective function (4.1) and encoding the constraints with the augmented Lagrangian, we have

$$\begin{aligned} L_\rho(m,\eta,\nu) = \frac{1}{2}\sum_{i=1}^{n}(y_i-m_i)^T\Sigma^{-1}(y_i-m_i) + \frac{n}{2}\log|\Sigma| + \sum_{i<j} p_\gamma(\|\eta_{ij}\|,\lambda) \\ + \sum_{i<j}\nu_{ij}^T(m_i-m_j-\eta_{ij}) + \frac{\beta}{2}\sum_{i<j}\|m_i-m_j-\eta_{ij}\|^2. \end{aligned} \tag{4.2}$$

Let u, η, and ν be equal to the vectorization (stack the columns on top of each other) of U, E, and V, respectively, where U is the $n \times p$ matrix where the i-th row is μ_i, and E and V are $\binom{n}{2} \times p$ matrices with $E = [\eta_{ij}, i < j]^T$ and $V = [\nu_{ij}, i < j]^T$. Similarly, let Y be the $n \times p$ matrix where the rows correspond to the observations y_i, and let $y = \text{vec}(Y)$. Also, define $D = I_p \otimes \Delta$, which is $\binom{n}{2}p \times np$, and $W = \Sigma^{-1} \otimes I_n$, which is $np \times np$, where $\otimes$ denotes the Kronecker product. Recall from Section 3.2.2 that $\Delta = [(e_i - e_j), i < j]^T$, the $\binom{n}{2} \times n$ matrix composed of $n \times 1$ vectors e_i, in which the i-th element is 1 and the remaining elements are 0. Then we can re-write the objective function (4.2) as

$$\begin{aligned} L_\rho(u,\eta,\nu) = \frac{1}{2}(y-u)^T W(y-u) + \frac{n}{2}\log|\Sigma| + \sum_{i<j} p(\|\eta_{ij}\|,\lambda) \\ + \nu^T(Du-\eta) + \frac{\beta}{2}\|Du-\eta\|^2. \end{aligned} \tag{4.3}$$

The update for u is found in the usual manner by setting the derivative equal to zero and then solving. It yields

$$\hat{u} = (W + \rho D^T D)^{-1}[Wy + \rho D^T(\eta - \beta^{-1}\nu)].$$

The η_{ij} update depends on the choice of penalty function used in the objective function. If the L_1 norm is used, it reduces to an element-wise application of the η-update from the univariate case (see Section 2.3). Note that this only encourages merging of

particular dimensions at a time instead of the whole vector. Thus, we use the L_2 norm to ensure entire vector differences are shrunk to zero. Let $\delta_{ij} = \mu_i - \mu_j + \rho^{-1}\nu_{ij}$. If we use the L_2 penalty function, $p(\|\eta_{ij}\|, \lambda) = \lambda\|\eta_{ij}\|$, then the update is

$$\hat{\eta}_{ij} = \mathrm{ST}_{\mathrm{block}}\Big(\delta_{ij}, \frac{\lambda}{\rho}\Big),$$

the block-wise soft thresholding operator, which can be expressed as

$$\begin{aligned}\mathrm{ST}_{\mathrm{block}}\Big(\delta_{ij}, \frac{\lambda}{\rho}\Big) &= \Big(1 - \frac{\lambda}{\rho\|\delta_{ij}\|}\Big)_+ \delta_{ij} \\ &= \begin{cases} 0 & \text{if } \|\delta_{ij}\| \le \frac{\lambda}{\rho} \\ \delta_{ij} - \lambda\dfrac{\delta_{ij}}{\|\delta_{ij}\|} & \text{if } \|\delta_{ij}\| > \dfrac{\lambda}{\rho} \end{cases}.\end{aligned}$$

If we use the MCP penalty, the update is

$$\hat{\eta} = \begin{cases} \dfrac{\mathrm{ST}_{\mathrm{block}}\Big(\delta_{ij}, \dfrac{\lambda}{\rho}\Big)}{1 - \dfrac{1}{\rho\omega}} & \text{if} \|\delta_{ij}\| \le \omega\lambda \\ \\ \delta_{ij} & \text{if} \|\delta_{ij}\| > \omega\lambda. \end{cases}$$

These updates can be derived by similar arguments used in Section 2.3. The Lagrange multiplier update is the same as (2.11).

The update for Σ is found in the usual way by setting the derivative equal to zero. It yields the sample covariance matrix:

$$\hat{\Sigma} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{m}_i)(y_i - \hat{m}_i)^T. \tag{4.4}$$

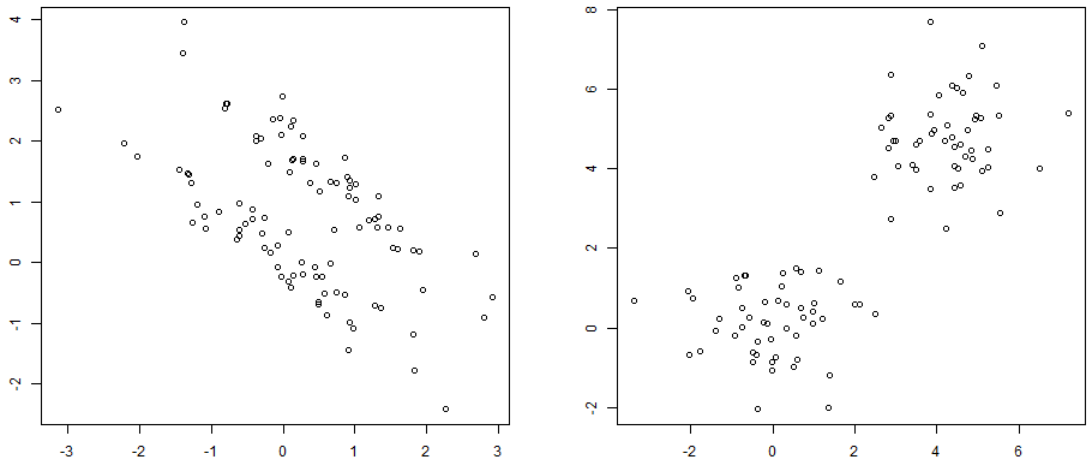
We can simply append this update to Step 1 in the ADMM iterates, updating $\hat{\Sigma}$ immediately after the location parameters are updated. However, when the penalty parameter λ is small,

or when the ADMM algorithm is initialized with the random sample itself, the $\hat{m}_i$ will likely be close to the y_i, especially during the first iterates of the algorithm. This leads to an unstable and insensible estimate of Σ. This obstacle becomes particularly apparent when constructing a solution path beginning with very small λ values. Another strategy is to set $\Sigma = I$ the identity matrix at the start of the algorithm, and then if enough shrinkage has occurred as the algorithm progresses, we can begin updating $\hat{\Sigma}$ with (4.4). However, it is difficult to define precisely what "enough shrinkage" should mean. For example, the covariance estimate when the current number of clusters is dramatically different than the true number of clusters will not capture much useful information. Furthermore, starting with $\Sigma = I$ encourages the formation of spherical clusters instead of trying to learn the correlation structure the data actually exhibit. Finally, note that the penalty term also penalizes all directions equally, which may be inappropriate of there is much more variability in one direction than in another. In general, learning and estimating the covariance structure is a very difficult problem in this context of over-parametrization where each y_i has its own m_i.

Assume for now that Σ is known. Then we can simply run the ADMM algorithm on the transformed data $y_i^* = \Sigma^{-\frac{1}{2}} y_i$ for each i, and set $\hat{\Sigma} = I$ the identity matrix in the ADMM iterates. However, this is actually minimizing a objective function that is different from (4.1). Let m_i^* denote the estimates of the transformed data y_i^*. This changes the penalty term in (4.1) to

$$\sum_{i<j} p_\omega(\left\|m_i^* - m_j^*\right\|, \lambda) = \sum_{i<j} p_\omega(\left\|\Sigma^{-\frac{1}{2}}(mu_i - m_j)\right\|, \lambda).$$

Thus, the covariance is present in the goodness-of-fit term but also in the penalty term. This



(a) Untransformed Data. $\mu_1 = (0,0)^T,\ \mu_2 = (1,1)^T$. (b) Transformed Data. $\mu_1^* = (0,0)^T,\ \mu_2^* = (4.47, 4.47)^T$

Figure 4.1: Plot of sample data Y and $Y^* = \Sigma^{-\frac{1}{2}} Y$.

encourages shrinkage of *within* cluster $\hat{\eta}_{ij}$ estimates and discourages shrinkage of *between* cluster $\hat{\eta}_{ij}$ estimates when the estimates are close to the truth. In Figure 4.1a, we want to penalize more strongly along the $y = -x$ direction, and less strongly along the $y = x$ direction. Otherwise, the true cluster centers may get shrunk together. We can accomplish this through transforming the data as shown in Figure 4.1b. In that case, we penalize all directions equally, and so clearly the true subgroups can be recovered along the solution path.

However, our original problem still remains: how can we obtain a first estimate of $\hat{\Sigma}$ to be able to transform the data? The transformation will be helpful if we have a good estimate of the covariance. We might obtain this by running the ADMM algorithm with

$\Sigma = I$, and then after convergence, use estimator (4.4). However, this still suffers from the weakness that it will encourage more or less spherical cluster shapes and underestimate the correlation present in the data. We also could use the MCLUST estimate after constraining all the Σ's to be equal. Since a good estimate of Σ is so crucial in this transformation approach, it remains a research goal to find a suitable covariance estimation strategy in this context. Another extension includes allowing multiple Σ_j's instead of assuming one common Σ, which leads to a goodness-of-fit term

$$L_n(m) = \frac{1}{2}\sum_{i=1}^{n}(y_i - m_i)^T\Sigma_i^{-1}(y_i - m_i).$$

However, it is unclear how to group and estimate the Σ_i, the number of parameters grows fast compared to the sample size. Note that this extension was already mentioned in Pen, Shen, and Liu (2013).

4.2 Multivariate Robust Loss

Recall our notation that observations $y_1, y_2, \ldots, y_n$ are of dimension d, where each y_i is associated with its own location vector m_i. Let M be the $n \times p$ matrix comprised of the m_i as row vectors. Let $\|a\| = \left(a^T a\right)^{1/2}$ be the Euclidean norm for any vector a. To introduce our proposed objective function, consider first minimizing $\sum_{i=1}^{n}\|y_i - m\|$ over m. If the y_i are of dimension one, then we obtain the absolute loss function and the usual median estimator. Using the (un-squared) Euclidean norm is a generalization of the median to more dimensions. It is sometimes called the geometric median or spatial median. See Aftab, Hartley, and Trumpf (2015) for an interesting paper on this kind of minimization problem.

We propose the following multivariate robust objective function:

$$Q(M, \lambda) = \sum_{i=1}^{n} \|y_i - m_i\| + \sum_{i<j} p_\omega\big(\|m_i - m_j\|, \lambda\big). \tag{4.5}$$

Since this is a natural generalization of the univariate median, we can use the same techniques used in Chapter 3 to compute the estimator. It is straightforward to derive the IRLS-ADMM updates here since many of the them are the same as before. We mainly need to discuss the m_i update, which we recall is a WLS update.

It is no surprise that the issues that obtained in the univariate case are also present here. Recall that for some loss function $h(\cdot)$, the weight formula can be expressed as

$$w_i = \frac{h'(y_i - m_i)}{(y_i - m_i)}. \tag{4.6}$$

Taking $h(e) = \|e\|$, we have

$$w_i = \frac{1}{\|y_i - m_i\|}, \tag{4.7}$$

which is defined for $m_i \neq y_i$ and undefined if $m_i = y_i$. Note that from an algorithmic point of view, the singularity when $m_i = y_i$ is a removeable one (Aftab, Harley, and Trumpf, 2015). However, we adopt the same approach used in Section 3.3; that is, we replace (4.7) with

$$w_i = \frac{1}{\max\{\, r, \|y_i - m_i\| \,\}}, \tag{4.8}$$

where $r > 0$ is a regularization parameter chosen to be small, such as $r = 0.0001$. As before in the univariate case, the r parameter here can be interpreted as the threshold parameter in a multivariate analog of the Huber loss function (Peker and Wiesel, 2016), which can be

written

$$h(y_i - m_i, r) = \begin{cases} \frac{1}{2}\|y_i - m_i\|^2 & \text{if} \quad \|y_i - m_i\| \leq r \\ r\|y_i - m_i\| - \frac{1}{2}r^2 & \text{if} \quad \|y_i - m_i\| > r. \end{cases} \tag{4.9}$$

This leads to a weights formula

$$w_i = \begin{cases} 1 & \text{if} \quad \|y_i - m_i\| \leq r \\ \frac{r}{\|y_i - m_i\|} & \text{if} \quad \|y_i - m_i\| > r. \end{cases} \tag{4.10}$$

Notice that the weight formulas (4.7) and (4.10) are equivalent since the weights only need to be known up to a common constant. We use the multivariate Huber loss with a small r as a smooth approximation to the Euclidean loss. The same idea can be found in Kärkkäinen and Äyrämö (2005) and a close cousin can be found in Fountoulakis and Gondzio (2016)

4.3 More Simulation Studies and Real Data Performance

We conducted some simulation studies in the multivariate setting to demonstrate the performance of our proposed method; that is, that solution path clustering with robust loss, as opposed to the usual least squares loss, improves the search for clusters. We also apply the methods to two real data sets. We evaluate performance using the same criteria that we used in Section 3.5, namely, the MSE, AAD, Rand Index, Viability rate, and BIC. Refer to the discussions in Section 3.5 for explanations of the criteria. Note that the Rand Index calculation remains the same, but the MSE and AAD calculations require a multivariate analog. Let there be K clusters with true cluster locations α_j, $j = 1, 2, \ldots, K$ of dimension d. If the clustering method is able to find a K-component solution with cluster location estimates $\hat{\alpha}_j$, then we calculate

$$\text{MSE} = \frac{1}{Kd}\sum_{j=1}^{K}\sum_{z=1}^{d}(\alpha_{jz} - \hat{\alpha}_{jz})^2 \quad \text{and} \quad \text{AAD} = \frac{1}{Kd}\sum_{j=1}^{K}\sum_{z=1}^{d}|\alpha_{jz} - \hat{\alpha}_{jz}|.$$

Recall that for the BIC comparisons, we need to include the full likelihood (integration constants and all) in order to keep the different loss function choices comparable. When the Huber loss function is used (and the Huber approximation to absolute loss), we needed to derive the "Huber density" as in Section 3.5. Unfortunately, my lack of mathematical fluency and sufficient time prevented me from deriving the multivariate Huber density. Instead, I used the multivariate Laplace likelihood in the BIC calculations. Note that in the univariate results, using the correct density for the Huber approximation to absolute loss improves the performance than when using the incorrect Laplace density, so we can reasonably expect that using the multivariate Huber density would improve the performance here as well.

We simulate a few different types of settings in an effort to explore where our method can perform well and when it begins to falter. Section 4.3.1 contains some straightforward settings, although some of the clusters overlap considerably. It also includes results for a null case (only 1 cluster exists) as well as a case using a larger number of clusters. Section 4.3.2 contains some settings where two dimensions have cluster information present and two dimensions are just noise (no cluster information present). In Section 4.3.3, we ran some simulations where the scales are different across the dimensions. We also included two scenarios with unequal mixing proportions. In Section 4.3.4, there are settings with an increasingly strong correlation present in the data. Even though the clusters are well-separated, the solution path clustering begins to fail when the correlation is strong. Finally,

we include some results on two real datasets.

4.3.1 Simulation Study 3

The first set of simulation scenarios we consider contain some straightforward cases of equal scale, no correlation, and equal mixing proportions. However, some of the cases have considerable overlap between the clusters. The errors e_{ij} in each dimension are simulated independently. We used $n = 200$ for all settings except for Setting 7 where $n = 500$. The settings are:

1. $P\big(m_i = (0,0)^T\big) = P\big(m_i = (\sqrt{8}, \sqrt{8})^T\big) = \frac{1}{2}$ with $e_{ij} \sim N(0,1)$.
2. $P\big(m_i = (0,0)^T\big) = P\big(m_i = (\sqrt{8}, \sqrt{8})^T\big) = \frac{1}{2}$ with $e_{ij} \sim t_2$.
3. $P\big(m_i = (0,0)^T\big) = P\big(m_i = (\sqrt{4.5}, \sqrt{4.5})^T\big) = \frac{1}{2}$ with $e_{ij} \sim N(0,1)$.
4. $P\big(m_i = (0,0)^T\big) = P\big(m_i = (\sqrt{4.5}, \sqrt{4.5})^T\big) = \frac{1}{2}$ with $e_{ij} \sim t_2$.
5. Five clusters with $P\big(m_i = (0,0)^T\big) = P\big(m_i = (\pm 5, \pm 5)^T\big) = \frac{1}{5}$ with $e_{ij} \sim N(0,1)$.
6. Five clusters with $P\big(m_i = (0,0)^T\big) = P\big(m_i = (\pm 5, \pm 5)^T\big) = \frac{1}{5}$ with $e_{ij} \sim t_2$.
7. Five clusters with $P\big(m_i = (0,0)^T\big) = P\big(m_i = (\pm 5, \pm 5)^T\big) = \frac{1}{5}$ with $e_{ij} \sim t_2$ and $n = 500$.
8. Null case: a single cluster in ten dimensions. Observations generated independently from Unif$[0,1]$ in each dimension. This is scenario (a) in Tibshirani, Walther, and Hastie (2001).

See Figure 4.2 for scatterplots of setting 1 and 5.

Table 4.1 contains the performance results for the H-LAD robust loss and the least squares loss. The first noticeable difference is that the robust loss has a much higher viability rate than the least squares. It is almost perfect across all the settings. Hence, the robust loss has an increased proclivity to find a solution containing the correct number of clusters, even in the presence of outliers where the least squares has more trouble. The robust loss also reports less bias than the least squares, even in some of the Gaussian error settings.

The BIC results are contained in Table 4.2. It appears that the results are somewhat sensitive to choice of c value, which is not something that we like to see. The robust loss selects the correct number of clusters a higher percentage of the time than does the least squares, but this is unsurprising since the robust loss also has higher viability rates. The only difference between Settings 6 and 7 is an increased sample size. Notice that the robust loss obtains a wider range of good c values while the least squares shows no improvement. It seems that the larger the sample size is, overall and per component, the more robust the BIC performance is to the choice of c value. Notice also that the BIC has no trouble in Setting 8, the null case of only one cluster.

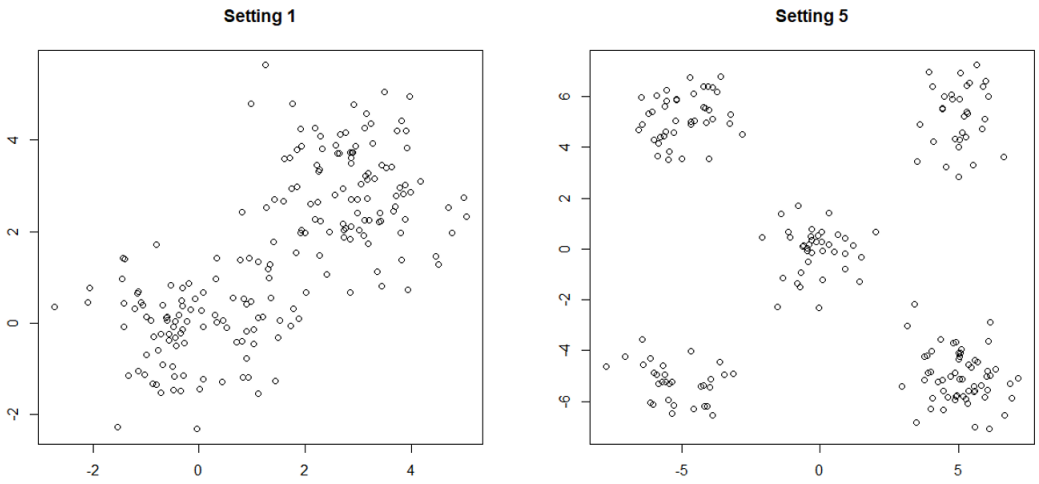


Figure 4.2: Scatter plots of settings 1 and 5 with $n = 200$.

		MSE(α)	AAD(α)	Rand	Viability Rate
Setting 1	H-LAD	0.0188	0.1119	0.9289	1
	LS	0.021	0.1156	0.9282	1
Setting 2	H-LAD	0.0304	0.1385	0.8791	1
	LS	186.6625	1.5704	0.8738	0.86
Setting 3	H-LAD	0.0329	0.1441	0.8299	1
	LS	0.0443	0.1686	0.823	1
Setting 4	H-LAD	0.0488	0.1726	0.8031	0.94
	LS	240.0538	2.2978	0.7773	0.75
Setting 5	H-LAD	0.0305	0.1417	0.9988	1
	LS	0.0237	0.1237	0.9991	1
Setting 6	H-LAD	0.0576	0.1908	0.9813	1
	LS	33.583	1.1097	0.9647	0.83
Setting 7	H-LAD	0.0221	0.1178	0.9811	1
	LS	4.8745	0.4250	0.9791	0.92
Setting 8	H-LAD	0.2498	0.4994	1	1
	LS	0.2499	0.4995	1	1

Table 4.1: Solution path clustering results, averaged across 100 replicates, for settings 1-7 with $n = 200$. An exception is Setting 7 where $n = 500$ and 50 replicates. "H-LAD" means Huber approximation to least absolute deviation and "LS" means least squares.

		c=5	c=10	c=15
Setting 1	H-LAD	1	1	0.36
	LS	0.03	0.81	0.98
Setting 2	H-LAD	0.99	0.98	0.11
	LS	0.02	0.14	0.3
Setting 3	H-LAD	0.98	0.57	0
	LS	0.02	0.74	0.96
Setting 4	H-LAD	0.94	0.43	0
	LS	0	0.13	0.29
Setting 5	H-LAD	1	0	0
	LS	0.93	1	1
Setting 6	H-LAD	1	0	0
	LS	0.13	0.37	0.61
Setting 7	H-LAD	1	1	0.36
	LS	0.06	0.14	0.30
Setting 8	H-LAD	1	1	1
	LS	1	1	1

Table 4.2: BIC results for settings 1-7 with $n = 200$ and 100 replicates. An exception is Setting 7 where $n = 500$ and 50 replicates. The c values are from $C_n = c \log\log n$ in the modified BIC. "H-LAD" means Huber approximation to least absolute deviation and "LS" means least squares.

4.3.2 Simulation Study 4

We explored a few simple simulation settings where two of the dimensions contain clustering information and two dimensions are simply noise, for a total of four dimensions. All three settings have the first two dimensions as coming from a 50/50 mix of two multivariate normal distributions with means $(0,0)^T$ and $(3,3)^T$ and identity covariance matrix. The noise dimensions simulated independent errors according to the following three settings:

1. *Normal Noise*: $e_i \sim N(0,1)$.

2. $\chi^2_{df=1}$ *Noise*: $e_i \sim \chi^2_{df=1}$.

3. $|t_2|$ *Noise*: $e_i \sim |t_2|$ the absolute value of a t_2 distribution.

See Figure 4.3 for scatter plots of settings 2 and 3. The solution path clustering performance results can be found in Table 4.3. For the first two settings, both methods do not have too much trouble, although some of the outliers have ruined the Rand Index from being close to perfect. The clusters in the first two dimensions are quite well-separated. The $|t_2|$ noise appears to have given the most difficulty, more so for the least squares loss. It appears that the bias must have been very large in several of the datasets. Also note that is has a lower viability rate than does the robust loss, showing that it could not even find a two cluster solution in 10 out of 100 datasets.

The BIC results are located in Table 4.4. The robust loss performs uniformly better than the least squares loss. The robust loss also is not sensitive to the choice of c value in these case. This is a good benchmark, since the clusters are quite well-separated in the first two dimensions, and the robust loss appears to exhibit robustness for the two

noise dimensions, unlike the least squares loss.

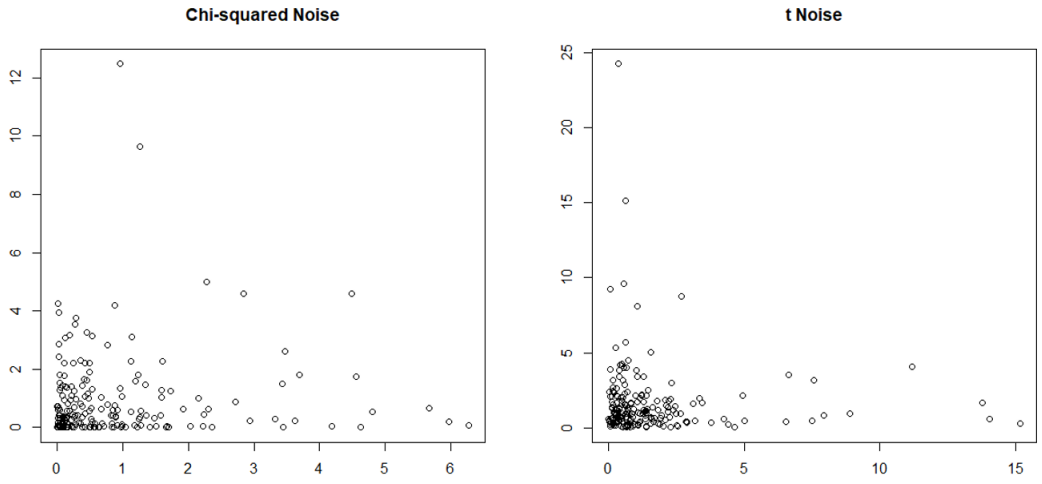


Figure 4.3: Scatter plots of noise dimension settings $\chi^2_{df=1}$ noise and $|t_2|$ noise, with $n = 200$.

		MSE(α)	AAD(α)	Rand	Viability Rate
Normal Noise	H-LAD	0.0163	0.0987	0.9222	1
	LS	0.0124	0.0887	0.9452	1
$\chi^2_{df=1}$ Noise	H-LAD	0.0586	0.2015	0.912	1
	LS	0.0313	0.1368	0.9168	0.99
$\lvert t_2 \rvert$ Noise	H-LAD	0.1041	0.2641	0.8837	0.99
	LS	78.8403	0.8575	0.8767	0.90

Table 4.3: Solution path clustering results, averaged across 100 replicates, for noise dimension settings 1-3 with $n = 200$. "H-LAD" means Huber approximation to least absolute deviation and "LS" means least squares.

		c=5	c=10	c=15
Normal Noise	H-LAD	0.89	1	1
	LS	0.05	0.93	1
$\chi^2_{df=1}$ Noise	H-LAD	0.87	1	1
	LS	0.01	0.14	0.57
$\lvert t_2 \rvert$ Noise	H-LAD	0.90	0.99	0.99
	LS	0.01	0.09	0.26

Table 4.4: BIC results for noise dimension settings with $n = 200$ and 100 replicates. The c values are from $C_n = c \log \log n$ in the modified BIC. "H-LAD" means Huber approximation to least absolute deviation and "LS" means least squares.

4.3.3 Simulation Study 5

We explored some simulation scenarios where both the scales are different and some of the mixing proportions are unequal. Let $N\big(m, \Sigma\big)$ represent the multivariate normal distribution with mean m and covariance matrix Σ. The settings are:

1. *Pi Letter*:

$$\frac{1}{3}N\Big((0,0), \text{diag}(0.05, 0.70)\Big) + \frac{1}{3}N\Big((1.5,0), \text{diag}(0.05, 0.70)\Big) + \frac{1}{3}N\Big((0.75,3), \text{diag}(0.70, 0.05)\Big).$$

2. *Mickey Mouse 1*:

$$0.6N\Big((0,0), I\Big) + 0.2N\Big((2.5,2.5), 0.2I\Big) + 0.2N\Big((-2.5,2.5), 0.2I)\Big).$$

3. *Mickey Mouse 2*:

$$0.70N\Big((0,0), 2I\Big) + 0.15N\Big((3,3), 0.05I\Big) + 0.15N\Big((-3,3), 0.05I)\Big).$$

See Figure 4.4 for scatter plots of all three settings. Note that the Mickey Mouse idea is not our own – we actually got the idea from the "k-means clustering" Wikipedia page, but a more appropriate citation for using Mickey Mouse data can be found in Agha and Ashour (2012).

The performance results can be found in Table 4.5. The robust loss is uniformly better across all criteria for the Pi Letter setting, most notably the viability rate. For the Mickey Mouse settings, both loss functions performed equally. However, we see that for Mickey Mouse setting 2, the more difficult of the two Mickey Mouses, both methods begin

to falter. Indeed, for about 30 datasets out of 100, the methods are not even able to find a three cluster solution. This Mickey Mouse is a more difficult setting with disparate scale sizes and mixing proportions. This is a limiting case where the implicit assumption of all equal scales is not proximate enough to the data structure.

Table 4.6 contains the BIC results. The robust loss method is only able to perform well for the Pi Letter and the easier Mickey Mouse setting, although it appears quite sensitive to the choice of c value. Both methods do not perform well for the more difficult Mickey Mouse, which is expected because of their lower viability rates. Interestingly though, for the Mickey Mouse setting 2, out of the 70 solution paths that contains a viable solution, the BIC with $c = 5$ was able to select 66 of them, a rate of 94%.

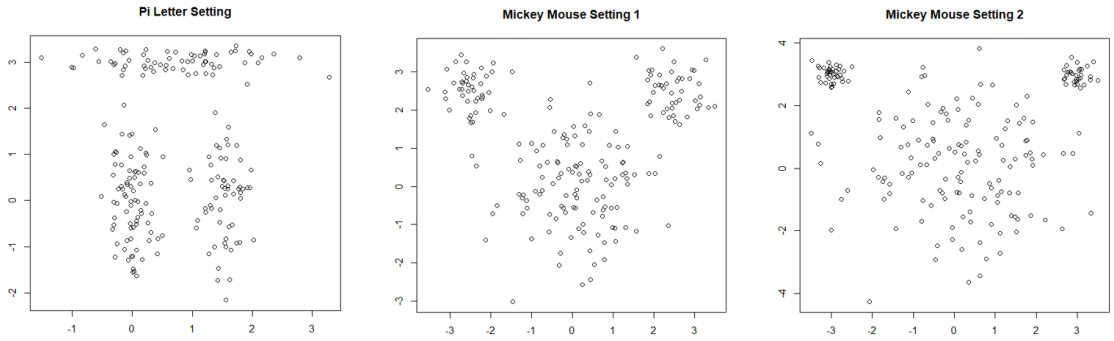


Figure 4.4: Scatter plots of the Pi Letter and Mickey Mouse settings with $n = 200$.

		MSE(α)	AAD(α)	Rand	Viability Rate
Pi Letter	H-LAD	0.0091	0.0647	0.9841	0.95
	LS	0.0335	0.0976	0.9714	0.74
Mickey Mouse 1	H-LAD	0.0096	0.0773	0.9629	0.99
	LS	0.0108	0.0791	0.9667	0.99
Mickey Mouse 2	H-LAD	0.0144	0.0819	0.9455	0.70
	LS	0.0517	0.1354	0.9329	0.73

Table 4.5: Solution path clustering results, averaged across 100 replicates, for the Pi Letter and Mickey Mouse settings with $n = 200$. "H-LAD" means Huber approximation to least absolute deviation and "LS" means least squares.

		c=5	c=10	c=15
Pi Letter	H-LAD	0.93	0.06	0
	LS	0.7	0	0
Mickey Mouse 1	H-LAD	0.96	0.19	0
	LS	0.08	0.87	0.99
Mickey Mouse 2	H-LAD	0.66	0.45	0
	LS	0	0.05	0.37

Table 4.6: BIC results for the Pi Letter and Mickey Mouse settings with $n = 200$ and 100 replicates. The c values are from $C_n = c \log\log n$ in the modified BIC. "H-LAD" means Huber approximation to least absolute deviation and "LS" means least squares.

4.3.4 Simulation Study 6

Finally, we explored some mild and strong correlation settings in order to find a limit of when our solution path clustering method begins to fail. These settings are of particular interest since our method implicitly assumes that the data do not exhibit any correlation structure.

1. *Mild Correlation 1*: $\frac{1}{2}N\big((0,0),\Sigma\big) + \frac{1}{2}N\big((\sqrt{8},-\sqrt{8}),\Sigma\big)$ where

$$\Sigma = \begin{bmatrix} 1 & 0.6 \\ 0.6 & 1 \end{bmatrix} \tag{4.11}$$

2. *Mild Correlation 2*: $\frac{1}{2}N\big((0,0),\Sigma\big)+\frac{1}{2}N\big((\sqrt{4.5},-\sqrt{4.5}),\Sigma\big)$ where Σ is given by (4.11).

3. *Strong Correlation 1*: $\frac{1}{2}N\big((0,0),\Sigma\big) + \frac{1}{2}N\big((\sqrt{4.5},-\sqrt{4.5}),\Sigma\big)$ where

$$\Sigma = \begin{bmatrix} 0.2 & 0.6 \\ 0.6 & 2 \end{bmatrix} \tag{4.12}$$

4. *Strong Correlation 2*: $\frac{1}{2}N\big((0,0),\Sigma\big) + \frac{1}{2}N\big((\sqrt{2},-\sqrt{2}),\Sigma\big)$ where Σ is given by (4.12).

See Figure 4.5 for a plot of these four correlation settings.

Table 4.7 contains the performance results. Both the robust loss and least squares perform quite well for the first three settings, with robust loss being only slightly better. Both methods begin to fail during Setting 4, the most difficult setting we consider here, which is the strongest correlation and the smallest distance between the cluster centers. Both methods have viability rates below 75%, and their Rand Indices are poor. One can imagine that the methods are trying to fit circular cluster shapes to the data, so that

each fitted cluster grabs about half the points from each true cluster. Clearly, the implicit assumption of an identity covariance matrix is not appropriate here. While it is maybe not surprising that violating the assumptions leads to a poor performance, it is still quite alarming since the two clusters are very clearly well-separated.

The BIC results are contained in Table 4.8. Both the robust loss and least squares loss are able to perform well for the first tw settings, althought interestingly they seem to prefer different c values. The least squares does not perform well for Setting 3 (for these c value choices) while the robust loss does great. Both methods do not perform well for the more difficult Setting 4, as would be expected from their lower viability rates. For $c = 15$, the least squares is able to get 67 correct out of 73 viable solutions, although the quality of the solution itself is likely not good, as evidenced by the low Rand Index in Table 4.7.

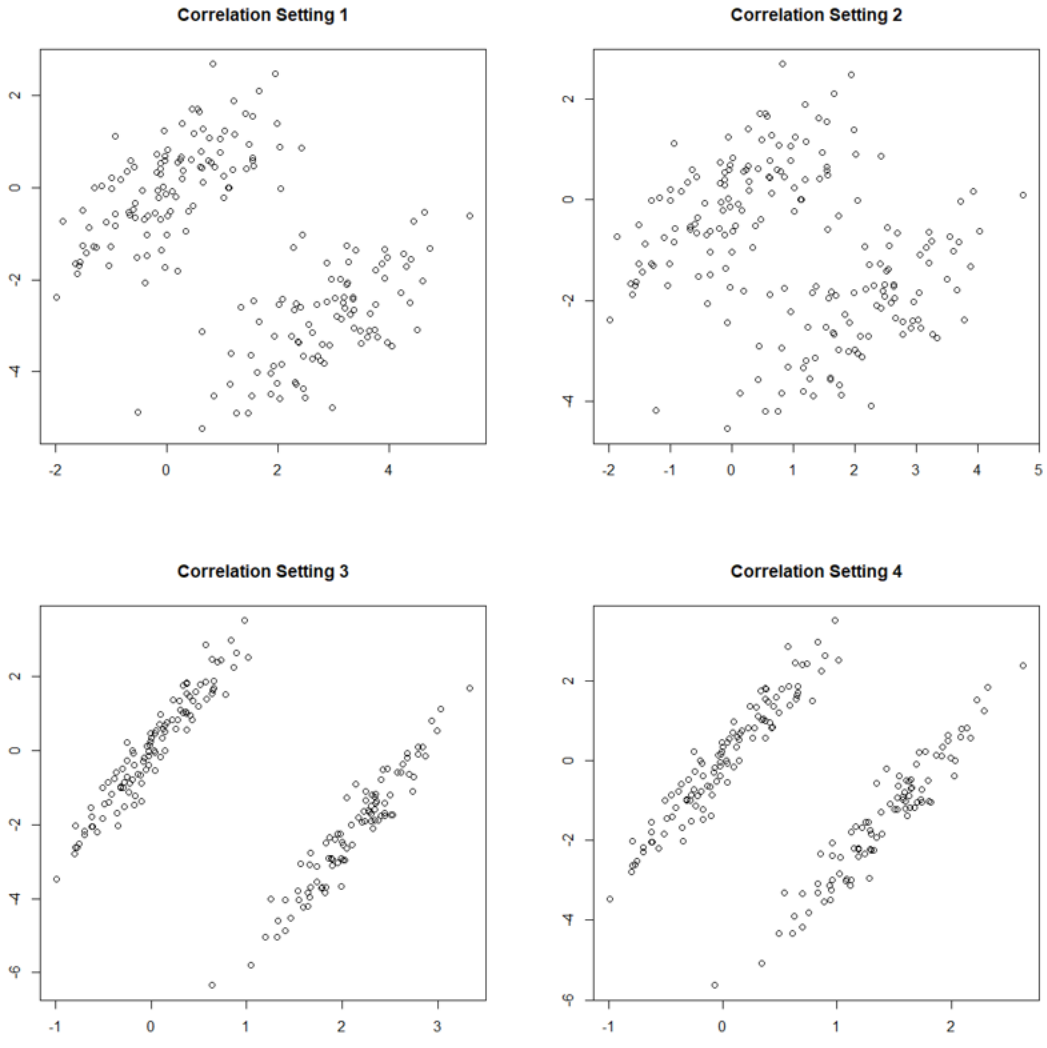


Figure 4.5: Scatter plots of the correlation settings with $n = 200$.

		MSE(α)	AAD(α)	Rand	Viability Rate
Setting 1	H-LAD	0.014	0.0961	0.9938	1
	LS	0.0106	0.0817	0.9963	1
Setting 2	H-LAD	0.016	0.1036	0.9606	1
	LS	0.0125	0.0878	0.9679	1
Setting 3	H-LAD	0.0185	0.0956	0.9965	0.98
	LS	0.0621	0.1441	0.9572	0.90
Setting 4	H-LAD	0.3164	0.4655	0.6145	0.74
	LS	0.3676	0.5312	0.5595	0.73

Table 4.7: Solution path clustering results, averaged across 100 replicates, for the correlation settings with $n = 200$. "H-LAD" means Huber approximation to least absolute deviation and "LS" means least squares.

		c=5	c=10	c=15
Setting 1	H-LAD	0.92	1	0.97
	LS	0	0.27	0.94
Setting 2	H-LAD	0.9	1	0.04
	LS	0	0.27	0.92
Setting 3	H-LAD	0.19	0.98	0.98
	LS	0	0.05	0.40
Setting 4	H-LAD	0.07	0.40	0
	LS	0	0.4	0.67

Table 4.8: BIC results for the correlation settings with $n = 200$ and 100 replicates. The c values are from $C_n = c \log \log n$ in the modified BIC. "H-LAD" means Huber approximation to least absolute deviation and "LS" means least squares.

4.3.5 Fisher's Iris Dataset

We applied our solution path clustering method, using the Huber approximation to absolute loss and the least squares loss, to the popular Fisher's Iris dataset. Note that this Iris dataset is usually automatically loaded in R, so that one can simply execute "plot(iris)" in the command line, without having to load or import anything, and a plot similar to Figure 4.6 will appear. Fisher's Iris data consists of 150 observations of three different species or subtypes of the Iris flower. There are 50 observations each of the three Iris subtypes setosa, versicolor, and virginca. There are four attributes recorded on each observation. They are the length and width of the petal, and the length and width of the sepal, measured in centimeters.

Figure 4.6 shows 6 different scatterplots corresponding to each pair of attributes, color coded according to the Iris subtype. It is clear that there are at least two natural clusters, with the black cloud of points comprising one cluster and the red and green cloud of points comprising the other. Whether there are really three natural clusters, corresponding to each color, is debatable. Following Pan, Shen, and Liu (2013), who also analyzed this data, we report cluster validation results using both two and three clusters as the "truth."

If we are trying to recover the labels corresponding to two clusters, both the Huber approximation to absolute loss and the least squares loss in the solution path clustering report perfect agreement; that is, their Rand indices are 1. That these two clusters are recovered can be regarded as a benchmark, since the two clouds of points are fairly well-separated. If we trying to recover the labels corresponding to three clusters, the Huber approximation to absolute loss has a Rand index of 0.8859 and the least squares loss has a

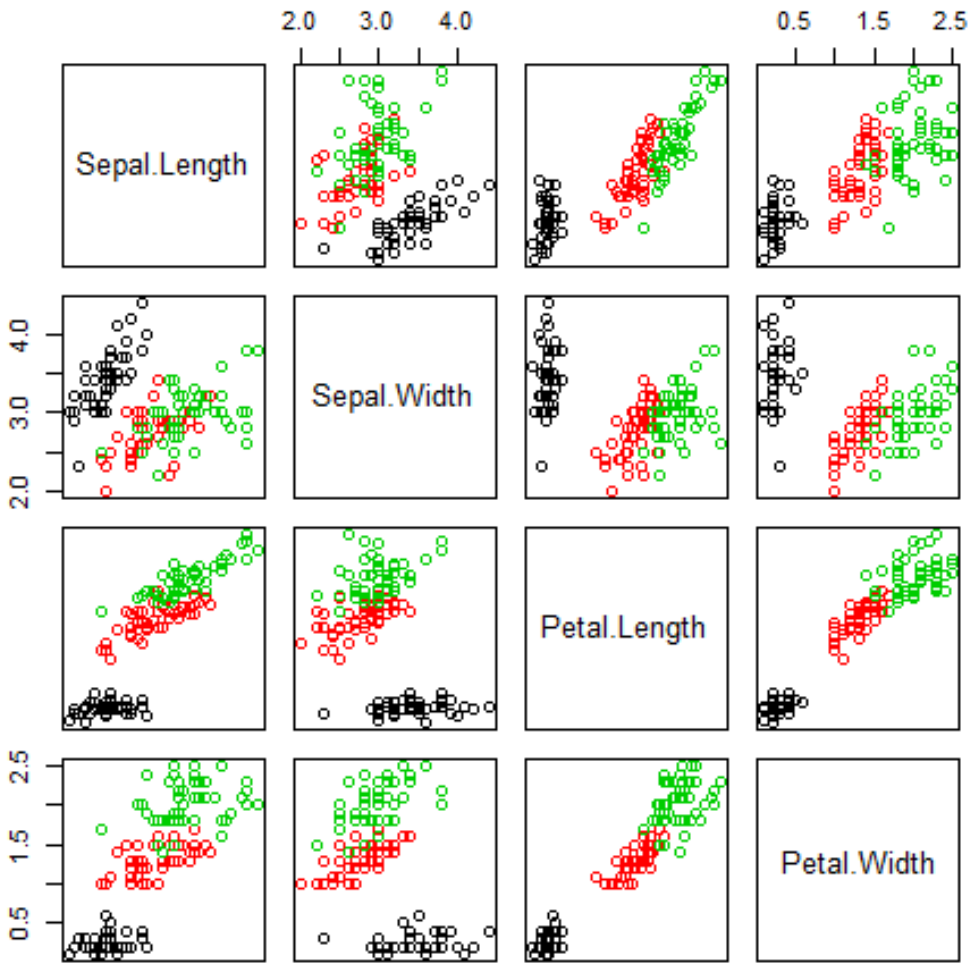


Figure 4.6: Plots of Fisher's Iris dataset where the three different colors correpsond to the three different Iris subtypes.

Rand index of 0.8797, very close to one another.

Fisher's Iris dataset is a great opportunity to discuss what cluster validation measures are appropriate, especially when real data with class labels are used. See Färber, et al. (2010) for an excellent explication of these issues. The ideas in this paragraph come from their paper. The issue can be summarized by the following quote: "Using classification data

	c=5	c=10	c=15
H-LAD	4	3	2
LS	3	2	2

Table 4.9: Number of clusters chosen by the modified BIC with different c values for the Iris Data Set.

for the purpose of evaluating clustering results, however, encounters several problems since the class labels do not necessarily correspond to natural clusters" (Färber, et al., 2010). The Iris data has exactly this problem, where there really only appears to be two natural clusters as opposed to three. This also emphasizes the difference between supervised and unsupervised learning. Supervised learning takes a training set where true class labels are available to learn how to predict future observations into a class. The whole point of unsupervised learning is to discover data structure that is not known to be there beforehand. The fact that a clustering scheme does not recover the true class labels can actually be useful and reveal new relationships between the classes, and it should be necessarily penalized for this. Interestingly, the BIC results in Table 4.9 are also uncertain about the number of clusters. Selecting four clusters seems to be undesirable, but besides that choice, the selections are between three clusters and two clusters.

At the very least, we need to be absolutely clear what the goal of the clustering is, such as discovering natural groupings (where "natural" could mean what the human eye would detect as distinct groupings, but even this definition causes some problems), or mode discovery, or even model recovery. As a detached researcher (as opposed to an

invested domain knowledge expert), the goal needs to be clear and fair in order to compare cluster results. Furthermore, cluster validation on real datasets also usually requires domain knowledge expertise for a proper interpretation.

4.3.6 Seeds Dataset

Consider the Seeds dataset from the UCI Machine Learning Repository. It consists of 210 observations with 7 recorded attributes of X-ray images of three different varieties of wheat: Rosa, Kama, and Canadian. There are 70 of each seed variety that were randomly selected for the experiment, of which we know the true labels. We can thus use the true labels as a kind of validation of the cluster partitions that result from our solution path clustering method. We remind the reader of the issues involved with using true labels as a cluster validation technique (see Section 4.3.5 for a discussion). In particular, is it not clear from the plot in Figure 4.7 that there are in fact three distinct clusters exist. Nevertheless, it is still useful to compare various three cluster partitions with the true labels.

Tables 4.10 and 4.11 shows some summary cluster results resulting from four different methods. Table 4.10 was copied directly from Charytanowicz, et al. (2010) and shows the results from a Complete Gradient clustering and a K-means clusters. Note that the Complete Gradient method and the K-means method are the only two discussed in their paper, so we imagine that those were the two best performing methods that they explored.

Table 4.11 shows the cluster summary results from our solution path clustering method using the Huber approximation to absolute loss and the least squares loss. Notice that the robust loss has a much smaller error rate than the least squares loss. Although the

robust loss solution path clustering has two more errors than the results from Charytanowicz, et al. (2010), it is very comparable.

We also included the BIC results for the Seeds dataset in Table 4.12. Only the robust loss at $c = 15$ correctly identifies three clusters, although at $c = 5$ it selected a huge number of clusters. This is undesirable since we would instead like to see the BIC not behave too sensitively to the choice of c.

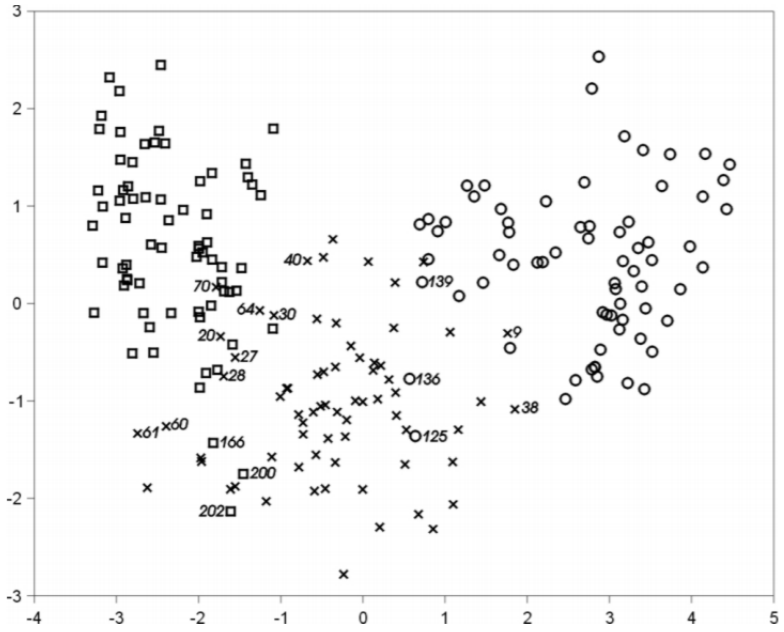


Figure 4.7: Plot of the two greatest principle components of the Seeds dataset. The circles are Rosa, the squares are Canadian, and the X's are Kama. This plot was taken from Charytanowicz, et al. (2010)

	Complete Gradient Clustering			K-Means Clustering		
Clusters	Correct	Incorrect	Total	Correct	Incorrect	Total
Rosa	67	2	69	66	2	68
Kama	59	6	65	65	12	77
Canadian	67	9	76	62	3	65
Total	193	17	210	193	17	21

Table 4.10: This is Table 1 from Charytanowicz, et al. (2010). The Complete Gradient clustering was their method of choice to compare to K-means clustering.

	Huber Approx. to AL			Least Squares Loss		
Clusters	Correct	Incorrect	Total	Correct	Incorrect	Total
Rosa	64	1	65	70	13	83
Kama	65	14	79	47	10	57
Canadian	62	4	66	60	10	70
Total	191	19	210	177	33	210

Table 4.11: Summary results for the Seeds dataset to compare Huber Approximation to Absolute Loss to the Least Squares Loss in Solution Path Clustering.

	c=5	c=10	c=15
H-LAD	171	6	3
LS	7	4	4

Table 4.12: Number of clusters chosen by the modified BIC with different c values for the Seeds Data Set.

4.4 Concluding Remarks

If computational speed and efficiency can be greatly improved, many more doors would open for potential applications of our solution path clustering method. For example, obtaining results for 100 replicates for setting 5 in Section 4.3.1 with 8-core parallel processing took over 10 hours, and setting 6 took almost 17 hours. There is a literature on analyzing the convergence of ADMM and AMA algorithms and accelerating their convergence speeds (for example, Goldstein, et. al., 2014). Another possibility is to code the algorithm in a faster language. In any event, if the computational speed was not an obstacle, larger sample sizes, larger number of dimensions, and re-sampling and bootstrap methods become much more feasible. Another way to improve computational efficiency is to not penalize *every* pairwise difference $m_i - m_j$, which grows at a fast rate of $\binom{n}{2}$. We could instead only penalize a subset of the pairwise differences. For example, we can use kNN weights so that each m_i is only connected to several of its close neighbors. This would also allow discovery of more exotic shaped clusters. This in itself constitutes an area of further research.

Another area of research is to find reliable methods to learn the covariance structure of the data. This is a difficult issue and none of the papers (that we know of) on solution path clustering offer any recommendations, although it seems quite a conspicuous issue.

More investigation into choosing the number of clusters is desirable. This issue in general has been a long-standing problem for any clustering method. It is difficult in part because the goal of clustering can vary from application to application. It is also sometimes

not quite clear what a "correct" number of clusters should even mean, since clustering is an unsupervised learning method. However, an automatic method to decide on the number of clusters can still be very useful. It seems that a specific strategy for solution path clustering can be developed, a strategy that explicitly incorporates the solution path in discerning the proper solution along the path. This is analogous to stopping strategies in hierarchical agglomerative clustering methods.

Our proposed solution path clustering method essentially consisted of modifying the loss function used to measure the goodness-of-fit. One can follow along this thread and derive solution path clustering for general likelihood models. One could also include covariates and construt a solution path for mixture regressions. In some preliminary work, we implemented some of these ideas, including using a Gamma likelihood, Poisson likelihood, a binomial likelihood, and mixture regression. In principle, the extensions are straightforward, but some obstacles that quickly arose included extreme sensitivity to initial values, difficulty in choosing good initial values, and getting stuck in local sub-optimal solutions. Some likelihood models are also inherently difficult. If are searching for groups of Poisson parameters, it becomes difficult to accurately classify the observations since the variance increases as the Poisson parameter increases. The Gamma likelihood is also difficult since each observation is parametrized by two parameters.

Our primary contribution is implementing robust loss functions in the solution path clustering framework. All of our other contributions and developments are companions to this extension. We developed the IRLS-ADMM algorithm to minimize our proposed objective function and proved its convergence to a local minimum. We extended some

of the theory of Ma and Huang (2017) to apply to our case of robust loss functions. We demonstrated the performance of our method through simulation studies and real data sets, including providing some preliminary results on selecting the number of clusters based on the modified BIC criterion (Wang, Li, and Leng, 2009). We also hope that the Literature Review we provide in Section 2.4 is useful since no such review of convex clustering and solution path clustering exists to our knowledge.

Besides the developments, the main goal of this dissertation is to show that using a robust loss function in the solution path clustering scheme improves the search for clusters. Robust loss functions are usually used to reduce bias in the estimation of certain parameters when outliers or extreme observations are present in the data. Estimation of location is typically not a priority in cluster analysis. However, the motivation of using robust loss in the solution path clustering scheme is to reduce the bias *within* the algorithms used to compute the estimates. This is similar to the reasoning behind using concave penalties to merge cluster locations. If the estimates are too biased in the iterations, the possibility of good cluster solution can be greatly diminished. That the robust loss improves the search for clusters, over and above the usual least squares loss, is evident from our simulation studies. For example, the viability rates for the robust loss functions are typically much higher than for the least squares. This means that the robust loss is able to find solution with the correct number of clusters much more often. This is not to claim that a correct number of solutions cannot be found by *some* λ value for the least squares loss, only that for the grid of λ values chosen, the robust loss is much more able to find them than the least squares loss. Some λ grid must be finally chosen to compute the solution path, and

the increased proclivity of the robust loss to discover the correct number of clusters is a definite strength.

On the other hand, it is important to note the limitations of our method. As presented in this dissertation, our method implicitly assumes that the data exhibit no correlation and each structure is on the same scale. We conducted simulation studies in order to find when this assumption breaks down. Particularly, when the correlation is strong, or when the different clusters are very different in size and membership, our method begins to fail. It can be disappointing because it can still fail in these cases even when the clusters are very clearly separated. Our method also performed badly on the exotic simulation settings described in Pan, Shen, and Liu (2013).

Overall, we recommend the Huber approximation to the absolute loss in the solution path clustering framework, along with the MCP penalty. It is robust to outliers, it does not require an estimate of a threshold parameter like other robust loss functions, it can approximate the absolute loss to almost arbitrary precision so that we can essentially interpret it as a median, its algorithmically stable compared to using the exact absolute loss, and its performance is quite consistently good in many settings.

Appendix A

A.1 Γ_2 Special case: $n = 4$

It is helpful to write out Γ_2 for the special case when $n = 4$ to help illustrate the development in the main body of the proof for Theorem 3. When $n = 4$, we have

$$\begin{aligned} P_n(m) = \lambda\big[&\rho(|m_1 - m_2|) + \rho(|m_1 - m_3|) + \rho(|m_1 - m_4|) \\ &+\rho(|m_2 - m_3|) + \rho(|m_2 - m_4|) + \rho(|m_3 - m_4|)\big] \end{aligned} \tag{A.1}$$

Calculating the gradient requires calculating the partial derivative with respect to each m_i. Recall that $\bar{\rho}(t) = \text{sign}(t)\rho'(|t|)$. We have

$$\begin{aligned} \frac{\partial}{\partial m_i}\rho\big(|m_i - m_j|\big) &= \Big[\frac{\partial}{\partial m_i}|m_i - m_j|\Big]\,\rho'\big(|m_i - m_j|\big) && \text{(chain rule)} \\ &= \text{sign}(m_i - m_j)\,\rho'\big(|m_i - m_j|\big) \\ &= \bar{\rho}(m_i - m_j), && \text{(A.2)} \end{aligned}$$

and

$$\begin{aligned}
\frac{\partial}{\partial m_j}\rho\big(|m_i - m_j|\big) &= \Big[\frac{\partial}{\partial m_j}|m_i - m_j|\Big]\,\rho'\big(|m_i - m_j|\big) && \text{(chain rule)}\\
&= -\text{sign}(m_i - m_j)\,\rho'\big(|m_i - m_j|\big)\\
&= \text{sign}(m_j - m_i)\,\rho'\big(|m_j - m_i|\big) && \text{(swap } m_i \text{ and } m_j\text{)}\\
&= \bar{\rho}(m_j - m_i). && \text{(A.3)}
\end{aligned}$$

Referring to (A.1), we have

$$\begin{aligned}
\frac{\partial P_n(m^a)}{\partial m_1} &= \lambda\big[\bar{\rho}(m_1^a - m_2^a) + \bar{\rho}(m_1^a - m_3^a) + \bar{\rho}(m_1^a - m_4^a) + 0 + 0 + 0\big],\\
\frac{\partial P_n(m^a)}{\partial m_2} &= \lambda\big[\bar{\rho}(m_2^a - m_1^a) + 0 + 0 + \bar{\rho}(m_2^a - m_3^a) + \bar{\rho}(m_2^a - m_4^a) + 0\big],\\
\frac{\partial P_n(m^a)}{\partial m_3} &= \lambda\big[0 + \bar{\rho}(m_3^a - m_1^a) + 0 + \bar{\rho}(m_3^a - m_2^a) + 0 + \bar{\rho}(m_3^a - m_4^a)\big],\\
\frac{\partial P_n(m^a)}{\partial m_4} &= \lambda\big[0 + 0 + \bar{\rho}(m_4^a - m_1^a) + 0 + \bar{\rho}(m_4^a - m_2^a) + \bar{\rho}(m_4^a - m_3^a)\big]
\end{aligned}$$

where we use (A.2) and (A.3). Notice that for each of the $\binom{n}{2}$ pairwise combinations of i and j, we have $\bar{\rho}(|m_i^a - m_j^a|)$ and $\bar{\rho}(|m_j^a - m_i^a|)$ appearing. To get to Γ_2, we need to multiply the i-th line above by $(m_i - m_i^*)$. Notice that each $(m_i - m_i^*)$ will appear in $n-1$ of the terms in each of the lines. Then Γ_2, which involves summing across these last four lines after multiplying the i-th line by $(m_i - m_i^*)$, can be written as

$$\Gamma_2 = \lambda \sum_{i<j} \bar{\rho}(|m_i - m_j|)(m_i - m_i^*) + \lambda \sum_{j<i} \bar{\rho}(|m_i - m_j|)(m_i - m_i^*)$$

which is how it appears in (3.59).

A.2 Γ_1 Special case: $n = 3$

The following shows an example calculation used in the development of Γ_1 in (**??**).

Consider the following:

$$\sum_{i=1}^{n} q_i(m_i - m_i^*) = \sum_{k=1}^{K} \sum_{i,j \in G_k} \frac{q_i(m_i - m_j)}{n_k} \tag{A.4}$$

Consider if there was only one subgroup k which has $n_k = 3$ elements with labels $i = 1, 2, 3$.

Concerning Equation (A.4), we have

$$\text{LHS} = q_1(m_1 - m_1^*) + q_2(m_2 - m_2^*) + q_3(m_3 - m_3^*)$$

Note that $m_1^* = m_2^* = m_3^*$ since they are all in the same subgroup. We also have

$$\begin{aligned} n_k \times \text{RHS} = {} & q_1(m_1 - m_1) + q_1(m_1 - m_2) + q_1(m_1 - m_3) \\ & + q_2(m_2 - m_1) + q_2(m_2 - m_2) + q_2(m_2 - m_3) \\ & + q_3(m_3 - m_1) + q_3(m_3 - m_2) + q_3(m_3 - m_3) \end{aligned}$$

One can think of the "rows" as index i and the "columns" as index j. Since the terms along the diagonal are zero, we are left with

$$\begin{aligned}
n_k \times \text{RHS} &= q_1(m_1 - m_2) + q_1(m_1 - m_3) \\
&\quad + q_2(m_2 - m_1) + q_2(m_2 - m_3) \\
&\quad + q_3(m_3 - m_1) + q_3(m_3 - m_2) \\
&= q_1(2m_1 - m_2 - m_3) \\
&\quad + q_2(2m_2 - m_1 - m_3) \\
&\quad + q_3(2m_3 - m_1 - m_2) && \text{(group like terms)} \\
&= q_1[3m_1 - (m_1 + m_2 + m_3)] \\
&\quad + q_2[3m_2 - (m_1 + m_2 + m_3)] \\
&\quad + q_3[3m_3 - (m_1 + m_2 + m_3)] && \text{(add zero inside each bracket)}
\end{aligned}$$

Dividing by n_k yields

$$\text{RHS} = q_1(m_1 - m_1^*) + q_2(m_2 - m_2^*) + q_3(m_3 - m_3^*)$$

which equals the LHS.

Now we illustrate why

$$\sum_{k=1}^{K} \sum_{i,j \in G_k} \frac{q_i(m_i - m_j)}{n_k} = \sum_{k=1}^{K} \sum_{i,j \in G_k} \frac{(q_j - q_i)(m_j - m_j)}{2n_k} \tag{A.5}$$

holds true in the special case of $n_k = 2$ and $i = 1, 2$ with only one subgroup. Concerning

Equation (A.5), we have

$$\begin{aligned} n_k \times \text{LHS} &= q_1(m_1 - m_1) + q_1(m_1 - m_2) + q_2(m_2 - m_1) + q_2(m_2 - m_2) \\ &= q_1(m_1 - m_2) + q_2(m_2 - m_1) \qquad \text{(first and last terms above are zero)} \\ &= q_1 m_1 - q_1 m_2 - q_2 m_1 + q_2 m_2. \end{aligned}$$

Similarly, the RHS only yields non-zero terms when $i \neq j$, so we have

$$2n_k \times \text{RHS} = (q_2 - q_1)(m_2 - m_1) + (q_1 - q_2)(m_1 - m_2) \tag{A.6}$$

$$= 2(q_1 m_1 - q_1 m_2 - q_2 m_1 + q_2 m_2). \tag{A.7}$$

Dividing by two yields equivalent between the RHS and LHS.

Bibliography

[1] Aftab, K., Hartley, R., and Trumpf, J. (2015), "Generalized Weiszfeld Algorithms for L_q Optimization," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37, 4.

[2] Agha, M. E., and Ashour, W. M. (2012), "Efficient and Fast Initialization Algorithm for K-means Clustering," *International Journal of Intelligent Systems and Applications*, 1, 21-31.

[3] Banerjee, A., and Ghosh, J. (2001), "Clickstream Clustering Using Weighted Longest Common Subsequences," in *Proceedings of the Web Mining Workshop at the 1st SIAM Conference on Data Mining*, pp. 33–40.

[4] Benaglia T., Chauveau D., Hunter D. R., and Young D. (2009), "mixtools: An R Package for Analyzing Finite Mixture Models," *Journal of Statistical Software*, 32(6), 1–29.

[5] Borman, S. (2006), "The Expectation Maximization Algorithm: A short tutorial,"

[6] Boyd, S., Parikh, N., Chu, E., Peleato, B., and Eckstein, J. (2011), "Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers," *Foundations and Trends in Machine Learning*, 3, 1-1122.

[7] Boyd, S., and Vandenberghe, L. (2004), *Convex Optimization*, Cambridge University Press. Available online for free.

[8] Boyd, S., and Vandenberghe, L. (2008), "Subgradients," Lecture Notes for EE364b, Stanford University, Winter 2006-07.

[9] Charytanowicz, M., Niewczas, J., Kulczycki, P., Kowalski, P. A., Łukasik, S., and Żak, S. (2010), "Complete Gradient Clustering Algorithm for Features Analysis of X-Ray Images," *Information Technologies in Biomedicine*, *Advances in Intelligent and Soft Computing*, 69, 15-24.

[10] Chang, L., Roberts, S., and Welsh, A. (2018), "Robust Lasso Regression Using Tukey's Biweight Criterion," *Technometrics*, 60(1), 36-47.

[11] Chen, J., and Chen, Z. (2008), "Extended Bayesian Information Criteria for Model Selection with Large Model Spaces," *Biometrika*, 95, 759-771.

[12] Chen, G. K., Chi, E. C., Ranola, J. M. O., and Lange K. (2015), "Convex Clustering: An Attractive Alternative to Hierarchical Clustering," *PloS Computational Biology*, 11:5, 1-31.

[13] Chen, W. C., and Maitra, R. (2011), "Model-Based Clustering of Regression Time Series Data via APECM – An AECM Algorithm Sung to an Even Faster Beat," *Statistical Analysis and Data Mining*, 4, 567–578.

[14] Chi, E. C., and Lange, K. (2015), "Splitting Methods for Convex Clustering," *Journal of Computational and Graphical Statistics*, 24, 994-1013.

[15] Coretto, P., and Hennig, C. (2010). "A Simulation Study to Compare Robust Clustering Methods based on Mixtures," *Advances in Data Analysis and Classification*, 4, 111-135.

[16] Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). "Maximum Likelihood from Incomplete Data via the EM Algorithm," *Journal of the Royal Statistical Society*. Series B (Methodological), Vol. 39, No. 1., pp. 1-38.

[17] Dolan, C. V., and Van der Maas, H. L. J. (1997), "Fitting Multivariate Normal Finite Mixtures Subject to Structural Equation Modeling," *Psychometrika*, 63, 227- 253.

[18] Eicker, F. (1963), "Asymptotic Normality and Consistency of the Least Squares Estimators for Families of Linear Regressions," *Annals of Mathematical Statistics*, 34, 447-456.

[19] Everitt, B. S. (1974), *Cluster Analysis*, John Wiley & Sons, Inc., New York.

[20] Everitt, B. S., Landau S., Leese, M., Stahl, D., (2011), *Cluster Analysis*, Wiley Publishing.

[21] Fan, J., and Li, R. (2001), "Variable Selection via Non-concave Penalized Likelihood and Its Oracle Properties," *Journal of the American Statistical Association*, 96, 1348-1360.

[22] Fang, Y., and Wang, J. (2012), "Selection of the Number of Clusters via the Bootstrap Method," *Computational Statistics and Data Analysis*, 56, 468-477.

[23] Fountoulakis, K., and Gondzio, J. (2016), "A Second-Order Method for Strongly Convex l_1-Regularization Problems," *Mathematical Programming*, A, 156:189-219.

[24] Fox, J., and Weisberg, S. (2013), "Robust Regression," `http://users.stat.umn.edu/ sandy/courses/8053/handouts/robust.pdf`.

[25] Fraley, C., and Raftery, A. E. (1998), "How Many clusters? Which Clustering Method? Answers via Model-Based Cluster Analysis," *The Computer Journal*, 41, 578-588.

[26] Fraley C., Raftery, A.E., Scrucca, L., Murphy, T. B., and Fop, M. (2017), "R Package 'mclust': Gaussian Mixture Modelling for Model-Based Clustering, Classification, and Density Estimation."

[27] Fu, F., and Zhou, Q. (2013), "Learning Sparse Causal Gaussian Networks with Experimental Intervention: Regularization and Coordinate Descent," *Journal of the American Statistical Association*, 108, 288-300.

[28] Goldstein, T., O'Donoghue, B., Setzer, S., and Baraniuk, R. (2014), "Fast Alternating Direction Optimization Methods," *SIAM Journal on Imaging Sciences*, 7(3), 1588-1623.

[29] Golub, G. H., Heath, M., and Wahba, G. (1979), "Generalized Cross-Validation as a Method for Choosing a Good Ridge Parameter," *Technometrics*, 21, 215-223.

[30] Gordon, A., (1999), *Classification*, Chapman and Hall, London.

[31] Hartley, R. I., and Zisserman, A. (2004), *Multiple View Geometry in Computer Vision*, 2nd edition, Cambridge University Press.

[32] Hartigan, J., (1975), *Clustering Algorithms*, Wiley, New York.

[33] Hayden, R. W., (2005), "A Dataset that is 44% Outliers," *Journal of Statistics Education*, 13(1), 1-12.

[34] He, X., and Shao, Q. M. (2000), "On Parameters of Increasing Dimensions", *Journal of Multivariate Analysis*, 73, 120-135.

[35] Hennig, C. (2010), "Methods for Merging Gaussian Mixture Components," *Advances in Data Analysis and Classification*, 4, 3–34.

[36] Hocking, T., Joulin, A., Bach, F., and Vert, J.-P. (2011), "Clusterpath: An Algorithm for Clustering using Convex Fusion Penalties," in *Proceeding of the Twenty Eighth International Conference on Machine Learning*, New York: ACM, 745-752.

[37] Holland, P. W., and Welsch, R. E. (1977), "Robust Regression Using Iteratively Reweighted Least-Squares," *Communications in Statistics - Theory and Methods*, 6:9, 813-827.

[38] Huber, P. J. (1964), "Robust Estimation of a Location Parameter," *Annals of Statistics*, 53, 73–101.

[39] Huber, P. J. (1981), *Robust Statistics*, New York, Wiley.

[40] Jain, A. K. and Dubes, R. C. (1988), *Algorithms for Clustering Data*, Prentice-Hall.

[41] Kaufman, L. and Rousseeuw, P., (1990), *Finding Groups in Data: An Introduction to Cluster Analysis*, Wiley, New York.

[42] Kärkkäinen, T., and Äyrämö, S., (2005), "On Computation of Spatial Median for Robust Data Mining," *Evolutionary and Deterministic Methods for Design, Optimization and Control with Applications to Industrial and Societal Problems.*

[43] Lange, K. (2004), *Optimization*, 1st edition, Springer, New York, NY.

[44] Li, J. Q., and Barron, A. R. (2000), "Mixture Density Estimation," Technical Report, Yale University.

[45] Lindsten, F., Ohlsson, H., and Ljung, L. (2011a), "Clustering Using Sum-of-Norms Regularization With Application to Particle Filter Output Computation," *IEEE Statistical Signal Processing Workshop*.

[46] Lindsten, F., Ohlsson, H., and Ljung, L. (2011b), "Just Relax and Come Clustering! A Convexification of K-means Clustering," Technical Report.

[47] Ma, S. and Huang, J. (2017), "A concave pairwise fusion approach to subgroup analysis," *Journal of American Statistical Association*, 112, 410-423.

[48] Marchetti, Y., and Zhou, Q. (2014), "Solution Path Clustering with Adaptive Concave Penalty," *Electronic Journal of Statistics*, 8, 1569-1603.

[49] McLachlan, G., and Peel, D. (2000), "Robust Mixture Modelling using the t-distribution," *Statistics and Computing*, 10(4), 339-348.

[50] Melnykov, V. (2016), "Merging Mixture Components for Clustering Through Pairwise Overlap," *Journal of Computation and Graphical Statistics*, Vol. 25, No. 1, pp. 66-90.

[51] Melnykov, V., and Maitra, R. (2010), "Finite mixture models and model-based clustering," *Statistics Surveys*, 4, pp. 80-116

[52] Mirkin, B. (2011), "Choosing the Number of Clusters," *WIREs Data Mining Knowledge Discovery*, 1(3), 252-260.

[53] Mordukhovich, B. S., and Nam, N. M. (2014), *An Easy Path to Convex Analysis and Applications*, Morgan and Claypool Publishers.

[54] Murtagh, F. and Conteras, P. (2011), "Algorithm for Hierarchical Clustering: An Overview," *Data Mining Knowledge Discovery*.

[55] Norets, A. and Pelenis, J. (2011), "Bayesian modeling of joint and conditional distributions," Unpublished manuscript, Princeton University.

[56] Pan W., Shen, X., and Liu, B. (2013), "Cluster Analysis: Unsupervised Learning via Supervised Learning with a Non-Convex Penalty," *The Journal of Machine Learning Research*, 14:1, 1865-1889.

[57] Parikh, N., and Boyd, S. (2013), "Proximal algorithms," *Foundations and Trends in Optimization*, 1, 3, 123-231.

[58] Peker, E., and Wiesel, A. (2016), "Fitting Generalized Multivariate Huber Loss Functions," *IEEE Signal Processing Letters*, 23, 11, 1647-1651.

[59] Pelckmans, K., De Brabanter, J., Suykens, J. A. K., De Moor, B., (2005), "Convex Clustering Shrinkage," *Statistics and Optimization of Clustering Workshop (PASCAL)*, London, U. K.

[60] Peña, J. M., Lozano, J. A., and Larraña, P. (1999), "An Empirical Comparison of Four Initialization Methods for the K-means Algorithm," *Pattern Recognition Letters*, 20:1027-1040.

[61] Rand, W. M. (1971), "Objective Criteria for the Evaluation of Clustering Methods," *Journal of the American Statistical Aassociation*, 66(336), 846-850.

[62] Razaviyayn, M., Hong, M., and Luo, Z. Q. (2013), "A Unified Convergence Analysis of Block Successive Minimization Methods for Non-Smooth Optimization," *SIAM Journal on Optimization*, 23:2, 1126-1153.

[63] Schork, N. J., Allison D.B., and Thiel, B. (1996), "Mixture Distributions in Human Genetics," *Statistical Methods in Medical Research*, 5, 155-178.

[64] Serfling, R. J. (1980), *Approximation Theorems of Mathematical Statistics*, New York, Wiley.

[65] Shah, S. A., and Koltun, V. (2017), "Robust Continuous Clustering," *Proceedings of the National Academy of Sciences of the United States of America*, 114(37), 9814-9819.

[66] Shen, X., Pan W., and Zhu, Y. (2012), "Likelihood-based Selection and Sharp Parameter Estimation," *Journal of the American Statistical Association*, 107, 223-232.

[67] Schwarz, G. (1978). "Estimating the Dimension of a Model," The Annals of Statistics, 6(2), 461-464.

[68] Stefanski, L. A., and Boos, D. D. (2002), "The Calculus of M-Estimation," *The American Statistician*, 56, 1, 29-38.

[69] Tan, K. M., and Witten, D. (2015), "Statistical Properties of Convex Clustering," *Electronic Journal of Statistics*, 9, 2324-2347.

[70] Tibshirani, R. (1996), "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistics Society*, 58, 267–288.

[71] Tibshirani, R., Walther, G., and Hastie, T. (2001), "Estimating the Number of Clusters in a Data Set via the Gap Statistic," *Journal of the Royal Statistical Society, Series B*, 63(2), 411-423.

[72] Tseng, P. (1991), "Applications of a Splitting Algorithm to Decomposition in Convex Programming and Variational Inequalities," *SIAM Journal on Control and Optimization*, 29, 119-138.

[73] Tseng, P. (2001), "Convergence of a Block Coordinate Descent Method for Nondifferentiable Minimization," *Journal of Optimization Theory and Applications*, 109, 475-494.

[74] Tukey, J. W. (1960), "A Survey of Sampling From Contaminated Distributions," *Contributions to Probability and Statistics*, 2, 448–485.

[75] Tzanetakis, G., and Cook, P. (2002), "Music Genre Classification of Audio Signals," *IEEE Transactions on Speech and Audio Processing*, 10, 293–302.

[76] Vattani, A. (2009), "K-means Requires Exponentially Many Iterations Even in the Plane," In *Proceedings of the 25th Annual Symposium on Computational Geometry (SCG)*, Aarhus, Denmark.

[77] Wang, Q., Gong, P., Chang, S., Huang, T. S., and Zhou, J., (2016), "Robust Convex Clustering Analysis," *IEEE 16th International Conference on Data Mining*.

[78] Wang, H., Li, B., and Leng, C. (2009), "Shrinkage tuning parameter selection with a diverging number of parameters," *Journal of Royal Statistical Society*, 71, 671–683.

[79] Wang, B., Zhang, Y., Sun, W. W., and Fang, Y., (2018), "Sparse Convex Clustering," *Journal of Computational and Graphical Statistics*, 27:2, 393-403.

[80] Xu, S., Qiao, X., Zhu, L., Zhang, Y., Xue, C., and Li, L. (2016), "Reviews on Determining the Number of Clusters," *Applied Mathematics and Information Sciences*, 10(4), 1493-1512.

[81] Xu, R. and Wunsch, D., (2005), "Survey of Clustering Algorithms," *IEEE Transactions on Neural Networks*, 16 (3), 645-678.

[82] Ye, J. (1998), "On Measuring and Correcting the Effects of Data Mining and Model Selection," *Journal of the American Statistical Association*, 93, 120-131.

[83] Yohai, V. J., and Maronna R. A. (1979), "Asymptotic Behavior of M-Estimators for the Linear Model," *The Annals of Statistics*, 7, 2, 258-268.

[84] Yuan, M., and Lin, Y. (2006), "Model Selection and Estimation in Regression with Grouped Variables," *Journal of the Royal Statistical Society, Series B*, 68, 49-67.

[85] Zhang, C. (2010), "Nearly unbiased variable selection under minimax concave penalty," *The Annals of Statistics* 38, 894–942.

[86] Zhu, C., Xu, H., Leng, C., and Yan, S. (2014), "Convex Optimization Procedure for Clustering: Theoretical Revisit," In *Advances in Neural Information Processing Systems*.